NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A FORMAL MODEL OF THE MAC LAYER
OF AN IMPROVED FDDI PROTOCOL

by

Jose Luiz Timbo Elmiro

September, 1991

Thesis Advisor: G. M. Lundy

Approved for public release; distribution is unlimited.

T254696

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. / 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION N

11. TITLE (Include Security Classification)

A FORMAL MODEL OF THE MAC LAYER OF AN IMPROVED FDDI PROTOCOL

12. PERSONAL AUTHOR(S)
Jose Luiz Timbo Elmiro

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 9/90 TO 9/91	September 1991	163

16. SUPPLEMENTARY NOTAION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Protocol Formal Specification, FDDI MAC standard, Systems of Communicatin Machines, FDDI Throughput Improvement, Protocol Verification.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This research examines an improved FDDI protocol which ideally raises the network throughput from 100 to maximum of 300 Megabits per second. It develops the details of the protocol structure at the MAC layer ar provides a formal specification using a formal model for protocol specification called *Systems of Communicatin Machines*. The study investigates the MAC FDDI standard and conforms the improved protocol to th specifications of that document. The MAC protocol employs a Timed-Token Controlled Concurrent Access wit simultaneous transmission on the FDDI dual ring. Key characteristics of FDDI are maintained in the improve protocol. The formal specification enhances protocol interpretation and verification. It reduces protoc ambiguities and allows proofs for protocol verification and correctness. A formal specification of a real-wor network protocol contributes to multivendor interoperability achievement.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Advisor's Name G. M. Lundy	22b. TELEPHONE (Include Area Code): (408) 646-2904	22c. OFFICE SYMBOL: CS/Ln

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted; All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

Approved for public release; distribution is unlimited.

A Formal Model of the MAC Layer of an Improved FDDI Protocol

by

Jose Luiz Timbo Elmiro
Lieutenant, Brazilian Navy
B.S., Brazilian Naval Academy, 1979

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Robert B. McGhee, Chairman
Department of Computer Science

ABSTRACT

This research examines an improved FDDI protocol which ideally raises the network throughput from 100 to a maximum of 300 Megabits per second. It develops the details of the protocol structure at the MAC layer and provides a formal specification using a formal model for protocol specification called Systems of Communicating Machines. The study investigates the MAC FDDI standard and conforms the improved protocol to the specifications of that document. The MAC protocol employs a Timed-Token Controlled Concurrent Access with simultaneous transmission on the FDDI dual ring. Key characteristics of FDDI are maintained in the improved protocol. The formal specification enhances protocol interpretation and verification. It reduces protocol ambiguities and allows proofs for protocol verification and correctness. A formal specification of a real-world network protocol contributes to multivendor interoperability achievement.

c.1

TABLE OF CONTENTS

I. INTRODUCTION 1

A. THE MOTIVATION 1

B. THE SCOPE OF THESIS 2

C. THESIS ORGANIZATION 3

II. THE FIBER DISTRIBUTED DATA INTERFACE 5

A. HIGH PERFORMANCE MULTI-NODE NETWORK 5

1. The FDDI Token Ring Architecture 6

a. FDDI Network Configurations and Topologies 7

b. FDDI Optical Fiber 11

2. FDDI Standards and Their Relations 12

B. THE MEDIA ACCESS CONTROL (MAC) STANDARD 16

1. Facilities Specification 18

a. The FDDI Symbol Set 18

b. Formats of Protocol Data Units 22

c. Timers and Counts 24

2. Operation 26

a. FDDI Timed-Token Access Method 26

b. Frame and token transmission 29

c. Stripping 30

3. Service Specification 30

C. DESCRIPTION OF THE EXISTING FORMAL MODEL "SYSTEMS OF COMMUNICATING MACHINES" 35

D. PREVIOUS WORK ON IMPROVEMENT OF FDDI 37

1. Suggestions on Improvement of FDDI 37

2. Other Work on Improvement of FDDI 41

III. THE IMPROVED FDDI PROTOCOL 45

A. PROTOCOL DESCRIPTION 45

1. The Improved FDDI MAC/PHY Organization 46

2. Changes in the Original Protocol 49

3. Protocol Data Units (PDU) Formats 51

B. THE TIMED-TOKEN CONTROLLED CONCURRENT ACCESS . . 56

1. Algorithms to Generate Subtoken Duration (Class) 59

2. Destination Address Station Actions 64

3. Reconfiguration 65

IV. FORMAL SPECIFICATION 66

A. BENEFITS OF A FORMAL SPECIFICATION 66

B. MODELING THE IMPROVED PROTOCOL 69

1. Notational Conventions 69
2. The Communicating Machines 70
3. Modeling The Interface Operations 71
4. The MAC Receiver Operation and Specification 75
a. The MAC Receiver State Diagram and Transition Table . . 77
(1) The Destination Address Station Receives a LLC Frame with Subtoken 78
5. Mac Transmitter Operation and Specification 104
a. The MAC Transmitter State Diagram and Transition Table 104
(1) The DA Station Removes a Frame Being Received - no subtoken issued 105
(2) The DA Station Receives a Frame With Subtoken and Uses the Subtoken 105
(3) A Station Repeats the Frame Downstream on the Ring 105
(4) Reliability Maintained When Configuration Changes (THRU to WRAP) 106
(5) A Station Captures a Token, Performs Simultaneous Transmission, and Issues a new Token. 106
6. Local and Timer Data Type Specification 131

V. PROTOCOL VERIFICATION 139

A. VERIFICATION 139

B. PROTOCOL CORRECTNESS THROUGH PROOFS 140

1. Proof that an LLC Frame is Copied and an Acknowledgment is Sent 141

VI. CONCLUSIONS 145

A. REVIEW OF THE RESEARCH 145

B. ISSUES FOR FURTHER RESEARCH 147

LIST OF REFERENCES 149

BIBLIOGRAPHY 151

INITIAL DISTRIBUTION LIST 153

ACKNOWLEDGMENTS

I would like to thank my wife Terezinha for her encouragement and love. I would also like to thank her for the high quality drawings of state diagrams that she prepared for the thesis.

I would like to express my sincere thanks to G.M. Lundy, my thesis advisor, for his unfaltering guidance in the conception and preparation of this thesis.

I. INTRODUCTION

A. THE MOTIVATION

In recent years, the rapidly growing demand for transfer of a massive amount of data between computing devices has led to a great deal of work towards improving network performance. Data transfer rates in networks have evolved from Kilobits per second to rates of 10 Megabits per second in CSMA/CD networks. The rapid progress in the processing power of workstations, the use of fiber optics as transmission medium, and the increased user expectations for performance have spurred development of a new standard for local area networks which achieves rates of 100 Megabits per second. This standard, the Fiber Distributed Data Interface (FDDI) is a token-passing ring local area network recently developed by the American National Standard Institute (ANSI) Accredited Standards Committee (ASC X3T9.5). FDDI has a 100 Megabits per second capacity which enables it to meet the bandwidth requirements for many applications.

Originally, FDDI was proposed by the ANSI X3T9.5 as a backend network between mainframe computers and their peripherals. The volume of data being moved or stored has reached proportions that dictate development of a high-performance interconnection among these computing devices. In the course of its development, new demands were brought, which in turn were accommodated by the emerging FDDI. As a result, the committee expanded the scope of FDDI to emphasize its application as a high-speed backbone network interconnecting other heterogeneous, lower-speed local area

networks such as token ring 802.5 and CSMA/CD. With the proliferation of the new powerful workstation-based computing environment, large and growing use of FDDI networks is expected.

The new FDDI protocol can still be improved to achieve higher network throughput. Lundy [Ref. 11] identifies inefficient use of network resources as one problem faced by this protocol. Lundy proposes alternative transmission procedures to increase the total network throughput by a factor of three to four times that of FDDI.

B. THE SCOPE OF THESIS

The goal of this research is to examine the suggested alternative transmission procedures which ideally raise the utilization to 300 Megabits per second, develop the details of an improved Media Access Control (MAC) protocol which supports these procedures and specify this MAC protocol using a formal model. The study investigates the current MAC FDDI standard and conforms the improved MAC protocol to the specification contained in that document.

The new MAC will satisfy three basic requirements. First, it will allow simultaneous use of both rings. Second, it will free a ring segment from frame repetition allowing creation of ring disjoint partitions. Third, it will permit concurrent use of the partitioned dual ring segments by two transmitting stations. This improved protocol can ideally achieve maximum throughput of 300 Megabits per second in a dual ring attachment topology.

To achieve the desired improvement, changes in the method of access to the physical medium are needed. New protocol data unit formats and the algorithms to substantiate the changes in the proposed access method are important initial research problems which are addressed. Another challenging problem in a dual ring operation is how to adapt the configuration function of FDDI when a node or link fails such that the same fault tolerance is maintained? Furthermore, since the new transmission procedures break the rings into disjoint partitions, how can the MAC supervisory frames circulate entirely in one logical ring during the initialization process? These and other intricate problems are analyzed and solved with the formal specification.

The specification in this thesis is a detailed formal protocol description which contributes to and enhances the standard document. First, it reduces documented protocol ambiguities; a desired feature in the interoperability achievement among multivendors. Next, it allows proofs for protocol correctness and development of protocol test procedures. Finally, it further increases understanding of the complex FDDI protocol. This thesis provides a formal specification of the improved FDDI protocol using a model called *systems of communicating machines* [Ref. 8]. This model was chosen because it is an effective tool for the specification of this protocol, providing flexibility as well as a formal basis for analysis.

C. THESIS ORGANIZATION

The thesis has six chapters. Chapter II reviews the FDDI network as a background for the thesis. The main focus is on the Media Access Control (MAC). Chapter III

introduces the MAC for the improved FDDI protocol. A timed-token controlled concurrent access is discussed and algorithms to generate subtoken are analyzed. Chapter IV provides the protocol MAC formal specification. Discussion will include the benefits of the formal specification, the communicating machines processes, and the interface operations. Chapter V provides proofs for correctness of protocol modules operation. Chapter VI concludes the thesis with a research review and provides suggestions for future work.

II. THE FIBER DISTRIBUTED DATA INTERFACE

This chapter provides the reader with information concerning the Fiber Distributed Data Interface (FDDI) network. The main emphasis of this chapter is on the Media Access Control (MAC) layer which will provide the background for the work developed in the subsequent chapters.

A. HIGH PERFORMANCE MULTI-NODE NETWORK

The Fiber Distributed Data Interface is a high-performance general purpose multi-station token ring network designed for efficient operation with a peak data transmission of 100 Megabits per second. FDDI provides many advantages over current LANs and as a high speed network it can meet the requirements of many applications.

The FDDI specification is a set of four standards being developed by a Task Group of Accredited Standards Committee (ASC) X3T9.5. The American National Standards for the physical layer PHY (ANSI X3.148-1988) and the Media Access Control (MAC) (ANSI X3.139-1987) have been approved and published. Other standards that will constitute the complete set are still actively being modified. In addition, the ISO/IEC JTC1/SC 13 standards committee are processing the FDDI documents as International Standards. The already approved documents constitute the basic FDDI. There are extensions to the basic FDDI now in the X3 approval process. [Ref. 19]

1. The FDDI Token Ring Architecture

An FDDI network consists of a set of nodes connected by optical transmission media into one or more rings. A ring is a closed loop of alternating nodes connected by the physical media. The data is transmitted from node to node serially over the ring as a stream of suitably encoded symbols. Each node regenerates and repeats the data downstream to the next node. The network may consist of hundreds of nodes, although no multivendor have built FDDI rings with a 200 nodes yet [Ref. 14].

The X3T9.5 committee took the existing IEEE 802.5 Token Ring protocol standard as the basis for development of FDDI. These two protocols are similar in functionality, however FDDI employs a timed token passing mechanism and uses fiber optics as transmission medium. Among other differences FDDI offers greater bandwidth and greater reliability. While 802.5 token ring operates with a maximum data rate of 4 Megabits per second FDDI provides for efficient network operation with a peak data rate of 100 Megabits per second. In fact, FDDI is the first standard designed for high performance general purpose multi-station network.

The token ring architecture of FDDI defines two rings. The first ring is called the primary ring and is used in the normal network operation. The second ring provides redundancy and is used only for reconfiguration of the network when a physical break occurs on the primary ring. This pair of rings forms the trunk ring of an FDDI network. The data in each ring flows in opposite directions. A timed-token method controls the access to the medium. Further discussion on the FDDI medium access method is given in the Media Access Control subsection of this chapter.

a. FDDI Network Configurations and Topologies

Figure 1 illustrates the types of stations and topologies used in FDDI networks. The figure shows the three types of stations: DAS, CON, and SAS, which are defined by the attachment to the ring. The Dual Attachment Station (DAS) is the basic building block of an FDDI network. It connects to a pair of physical links to carry signals in opposite directions. The Dual Attached Station has two ports A and B; one for each ring, and attaches directly into the trunk ring in a peer connection. A second type of station uses a concentrator (CON) as a device to provide the attachment. A concentrator has additional ports (master ports) beyond those required for its own attachment to the FDDI ring. Concentrators are either Single Attachment (SAC) or Dual Attachment (DAC). A dual attachment concentrator can attach directly to the trunk ring, and provide the capability to connect slave stations into either, or both, of the logical rings provided by the trunk ring. A third type of station is the Single Attachment Station (SAS), which has one port (slave port) and therefore would not attach directly into the trunk rings. Instead, a Single Attachment Station connects only to a concentrator. The Dual Attached Stations or Dual Attached Concentrators are also called Class A stations whereas the Single Attached Stations are called Class B stations. [Ref. 13]

FDDI allows only one trunk ring, however cascaded concentrators can attach multiple trees of varying levels. This topology is called a dual ring of trees. It combines the advantages of a dual ring with the advantages of a tree configuration.

One advantage of a dual ring design is its superior reliability provided by DAS and CON. For example, the dual counter-rotating ring alleviate the problem of

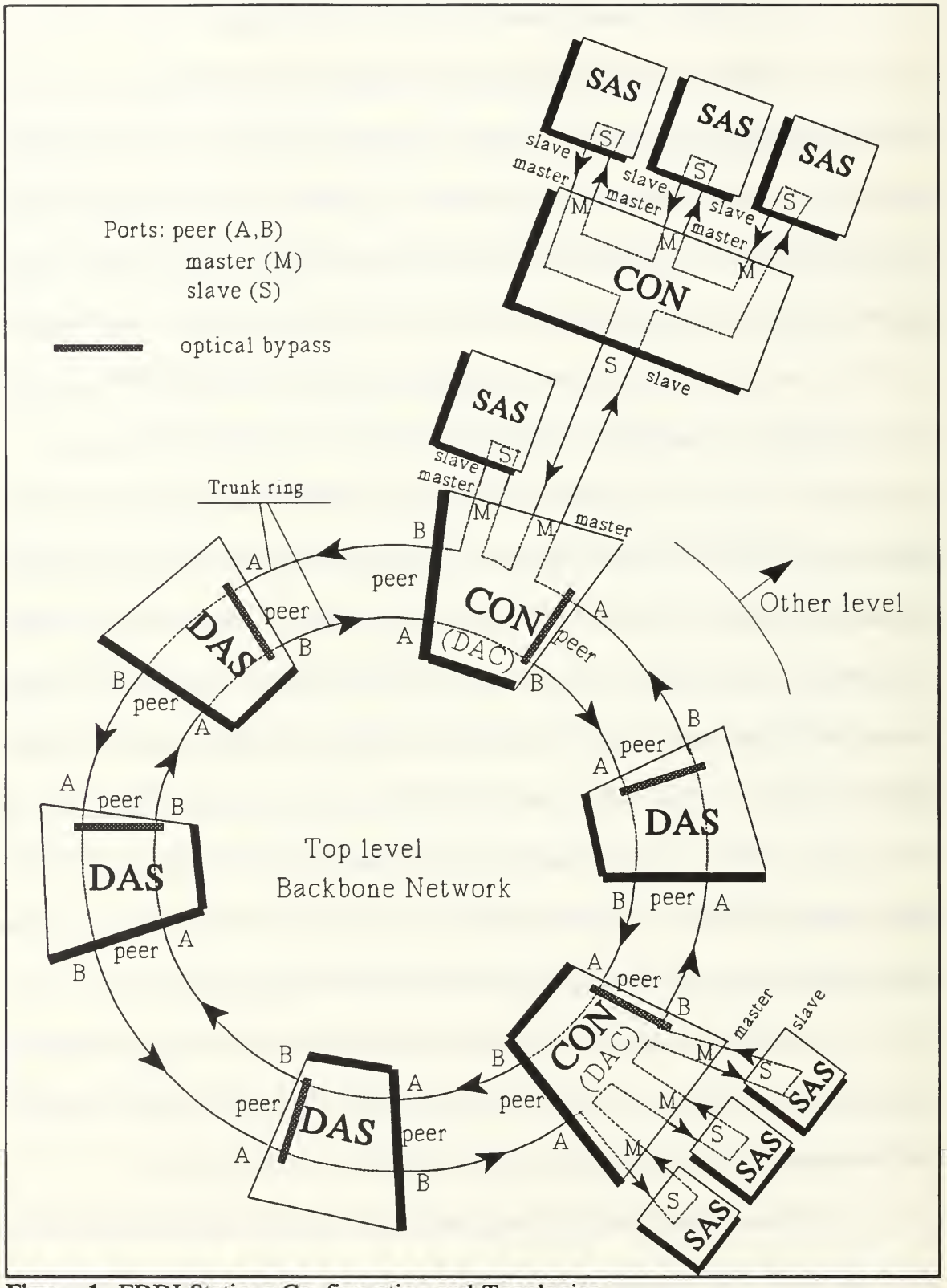


Figure 1: FDDI Stations Configuration and Topologies

multiple points of failure within the network. If a node or link fails, the two counter-rotating paths wrap together around the fault, allowing communication to continue. In this case the configuration changes to the Wrap mode. In the Wrap mode there is a single logical ring as opposed to the normal configuration or Thru mode, which has a dual logical ring. Also, DAS may offer the bypass capability by means of an optical switch. This mechanism is useful if a station for some reason is removed from the ring path. This feature allows the remaining stations to continue in the Thru mode. Figure 2 illustrates these configuration changes.

Another advantage of a ring design is that the optical fiber easily accommodates ring configurations. This approach significantly reduces the size, cost, and complexity of the hardware required by a network since the optical fiber medium offers high bandwidth which is best suited for bit-serial transmission. [Ref. 12]

The tree topology provides fault tolerance. For example, when removing a station or the cable connecting the SAS to the CON fails, the bypassing of the failure occurs electronically within the CON. In the case where there are many stations within a facility, CON allow for any number of such failures or disconnections without affecting the connectivity of all other stations on the FDDI network. Combining the dual ring and the tree portions in one topology, the resulting dual ring of trees provides a very high degree of fault tolerance and increases the availability of the backbone ring. [Ref. 7]

Essentially, the improvement in the protocol modelled in this thesis is achieved by the effective use of both rings during normal network operation. Therefore, the topology addressed by this protocol is a dual ring attachment configuration. However,

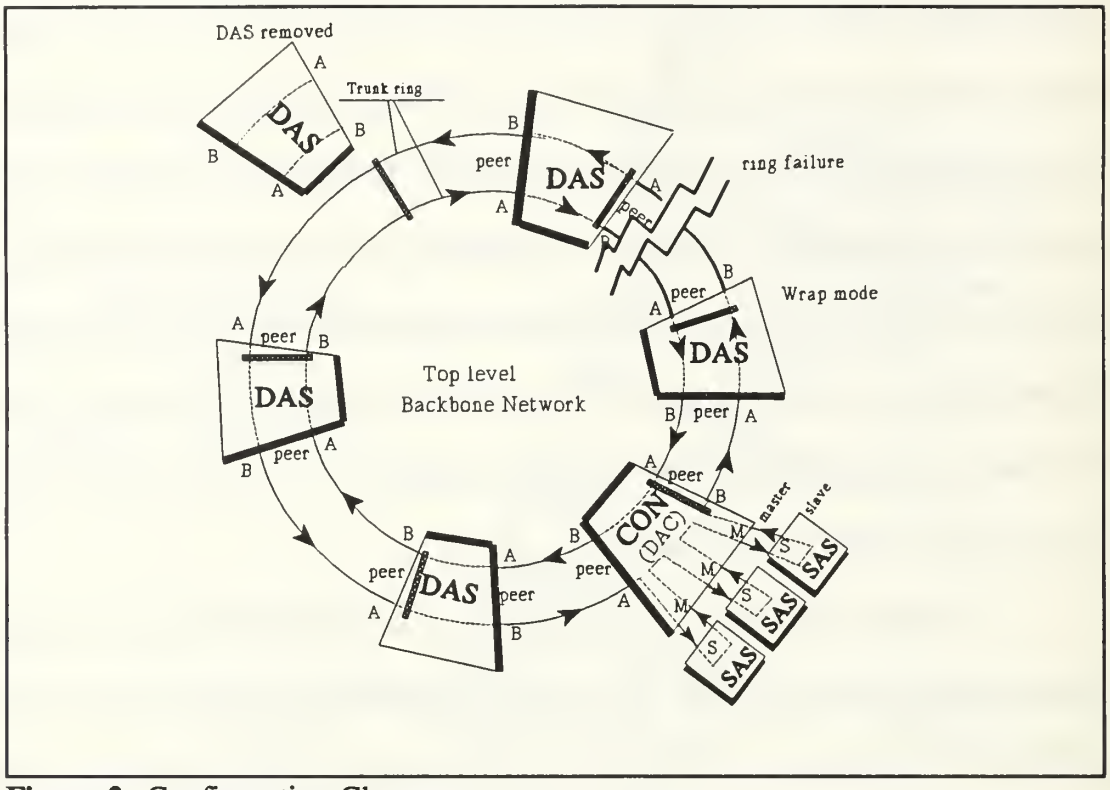


Figure 2: Configuration Changes

as the figure illustrates, other levels in the FDDI topology may use a single attachment type of station which could not be implemented with the improved protocol since the secondary ring for these stations is absent. This issue is discussed in Chapter III.

An upward-compatible version of the initial FDDI is FDDI-II. FDDI-II allows the creation of an integrated services LAN because it adds the capability for circuit switched services to the packet services of the basic FDDI. The concept behind FDDI-II is time-division multiplexing (TDM) sixteen separate channels, with each channel having a maximum of 98 Megabits per second full duplex. FDDI-II is intended for simultaneous voice, video, and data capabilities. This thesis uses the basic FDDI as the basis to model the improved FDDI protocol.

b. FDDI Optical Fiber

The basic FDDI standard uses graded-index multi-mode optical fiber with surface Light Emitting Diodes (LEDs) transmitting at a nominal wavelength of 1325 nanometers. This wavelength is a near "zero-dispersion" in conventional germanium-doped silica fibers. Multi-mode fibers can collect output from Light Emitting Diodes, which have much larger emitting areas, are less expensive, and have high reliability. The graded-index reduces the mode dispersion of a multi-mode fiber, allowing higher bandwidths. Commercial graded-index fibers attains bandwidths of around a Gigahertz-Kilometer [Ref. 6].

The FDDI standard recommends the use of 62.5/125 micron (one thousandth of a millimeter) optical fiber with minimum required fiber bandwidth is 500 Megahertz-Kilometer at the 1300 nanometer operating wavelength of transmitters and receivers. At this wavelength, the attenuation of multi-mode fiber is in the range of 0.6 to 1.0 decibel/kilometer. The 62.5/125 micron refers to the diameter of the core and cladding of the optical fiber. Alternates multi-mode fibers such as 50/125, 100/140, or 85/125 are also allowed [Ref. 17]. The fiber links in FDDI can be up to two kilometers apart. The standard specifies an instantaneous data transmission rate of 100 Megabits per second, which is the absolute upper bound on the throughput rate of the network. The effective sustained data rate at the data link layer can be over 95 percent of this peak rate [Ref. 13].

Single-mode fiber is another step in the evolution of FDDI. The maximum distance of two Kilometers achieved with multi-mode is extended to 60

kilometers in the specifications for the single-mode fiber added to the FDDI standards; however, the data rate is maintained at 100 Megabits per second. With larger distances FDDI can be an important backbone high-speed network to link other LANs.

2. FDDI Standards and Their Relations

Figure 3 depicts the organization of the FDDI standards. These standards relate to layer 1 (Physical) and layer 2 (Data Link) of the OSI reference model. The basic FDDI assumes the use of IEEE 802.2 standard, Logical Link Control (LLC), however does not specify this standard. The Basic FDDI is organized as follows:

- Physical Layer (PL), which is divided into two sublayers: the Physical Medium Dependent (PMD), and the Physical Layer Protocol (PHY).
- Data Link Layer (DDL), which is divided into sublayers: the Media Access Control (MAC), and the optional Logical Link Control (LLC). Other optional sublayers are being processed as an enhancement to the approved FDDI standard.
- Station Management (SMT), which conducts a node level control necessary to ensure cooperation with other nodes on a ring.

The arrows between the entities indicate their relations. FDDI describes these relations as services that two entities must provide and require at the interface between them.

The Physical Layer Medium Dependent standard (PMD) is the bottom sublayer of the Physical Layer. This sublayer provides all services necessary to pass successfully the serial bit stream of digital code from node to node. The PMD defines the physical medium, drivers and receivers, optical signal and waveform requirements, cable plant, connectors, and medium characteristics. This standard specifies an optical

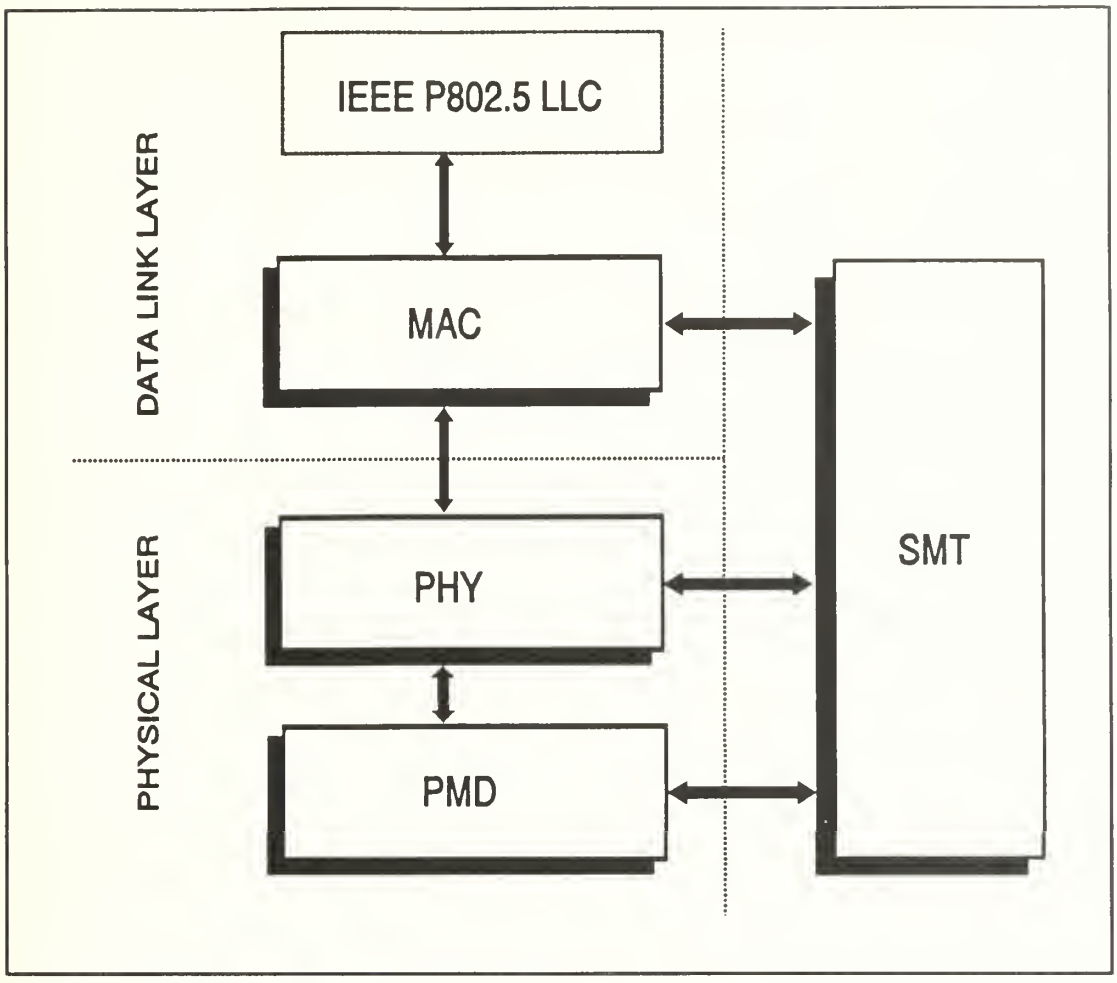


Figure 3: Organization of FDDI standards

multi-mode fiber ring for the basic FDDI with a transmission data rate of 100 Megabits per second, using the nonreturn to zero, invert on ones (NRZI) 4B/5B encoding scheme. The wavelength specified for data transmission is 1325 nanometers. The default values for nodes connectivity establish 1000 physical links as basis, a maximum distance between adjacent repeaters of two Kilometers, and 200 kilometers of total fiber path length. These values typically correspond to 500 nodes distributed over 100 kilometers of dual fiber cable; however, FDDI can support larger networks by increasing the node connectivity values.

The Physical Layer Protocol (PHY) defines the physical layer services and addresses the data encoding/decoding, clocking, latency, and data framing. The PHY defines the physical layer services in terms of primitives and parameters. These primitives support the transfer of data from a single MAC entity to all MAC entities contained within the same local network defined by the medium. The PHY specifies the data encoding scheme by using a code called 4B/5B. This scheme does encoding four bits at time; it encodes each four bits of data into a symbol with five cells such that each cell contains a single signal element (presence or absence of light). In effect, it encodes each set of four bits as five bits. Thus, the protocol achieves 100 Megabits per second with 125 Megabaud. The PHY further encodes the 4B/5B using Non Return to Zero Inverted (NRZI), which uses differential encoding, which improves reliability in the presence of noise and distortion. This reliability is because differential encoding decodes the signal comparing adjacent signal elements rather than the absolute value of a signal element.

The Media Access Control (MAC) corresponds to the lower half of the Data Link Layer for the FDDI. The standard assumes that MAC can be developed to operate under the Logical Link Control (LLC) of the ANSI/IEEE 802 series. It is the MAC which actually specifies the token passing and data transfer features, and thus is of primary interest in this thesis.

The Media Access Control standard presents its specifications in terms of the MAC services, facilities, and the MAC protocol operation. The number of MAC services depends on the implementation; however, a minimum set of services shall be provided to satisfy the requirements of the Logical Link Control or any other higher level protocol

being used. The interface includes facilities for transmitting and receiving protocol data units (PDU), and provides operation status information for use by higher-layer error recovery procedures. The MAC specification defines the frame structure and the interactions that take place between MAC entities. In general, MAC specifies access to the medium, addressing, data checking, frame format, and frame content interpretation.

The Station Management standard (SMT) defines the FDDI station configurations, the ring configurations, and specifies the control required for proper operation and interoperability of stations in an FDDI ring. FDDI divides SMT operation into three broad categories [Ref. 13]: Connection Management (CMT), Ring Management (RMT), and Operational Management. CMT establishes and maintains the physical and logical topology of the FDDI network and manages the physical layer resources of an FDDI node. CMT includes the protocols for ring formation and fault isolation on the duplex optical data links that connects FDDI stations. RMT deals with the correctness of the logical ring operation. It manages the MAC layer resources of a station. Operational Management deals with the management of the FDDI network in the operational state. These are multiple stations services with the purpose of proper operation and interoperability achievement.

There are also other standards being developed as extensions to the basic FDDI. A Single-Mode Fiber version of the PMD standard (SMF-PMD) will increase the length of permissible fiber links from two to 60 Km. This standard provides an alternate to the basic PMD. Another standard being developed that provides an alternate to basic PMD is the FDDI-to-SONET (Synchronous Optical NETwork) physical Layer Mapping

Function standard. This standard will provide a transport for FDDI over SONET common carrier facilities [Ref. 13]. A third extension to the basic FDDI is the Hybrid Ring Control standard (HRC), which specifies FDDI-II. HRC provides multiplexing of packet and circuit switched data on the shared FDDI medium. The purpose of FDDI standards is to ensure interoperability between conforming FDDI implementations. The implementations shall follow the guidelines of the standards functional descriptions, however these implementations may employ any design technique that is interoperable [Ref. 18].

B. THE MEDIA ACCESS CONTROL (MAC) STANDARD

The Media Access Control (MAC) provides deterministic access to the medium, address recognition, generation and verification of frame check sequences. Its primary function is the delivery of frames, including frame insertion, repetition, and removal. [Ref. 19]

As with IEEE 802.5, FDDI configures the network as a ring. The basic operation of the token is similar for both 802.5 and FDDI, however FDDI employs dual counter-rotating rings: a primary and a secondary ring. The secondary ring exists primarily for the purpose of redundancy. This improves reliability on an FDDI network.

The basic FDDI network employs two classes of services, synchronous service and asynchronous service. Synchronous service is for applications where the nodes deliver predictable units of data at regular intervals, such as real-time control that requires access to the channel within a specific time period [Ref. 4]. Each node is allotted a fraction of

the total available FDDI bandwidth for its synchronous service. Asynchronous service receives lowest priority. This service is permitted only after the station has finished its synchronous transmissions and if the timing requirements allow the service execution.

The FDDI MAC also provides a mechanism that satisfies the requirement for dedicated multiframe traffic. A station may initiate an extended dialogue requiring substantially all of the unallocated (asynchronous) ring bandwidth by using a restricted token. The initiating station captures a nonrestricted token, transmits the first frame of the dialogue to the addressed station, and then issues a restricted token. The destination address station receives the initial dialogue frame, enters the restricted mode, and then these two stations may exchange data frames and restricted tokens for the duration of the dialogue. Restricted token mode is terminated upon the capture of a restricted token by the terminating station. This station transmits its final dialogue frame, then issues a nonrestricted token. Any station may transmit synchronous frames upon capture of either type of token. [Ref. 19]

The FDDI MAC protocol has a number of other functions. MAC is responsible for data integrity. For example, it ensures the frames are not corrupted. A valid frame criteria defined in the MAC enforces the required reliability of frame reception. Another responsibility is data stripping. For example, the MAC of a transmitting station is responsible for the removal from the ring of all the frames that it has placed on the ring. Ring initialization, error detection and correction are also responsibilities of MAC. Ring initialization ensures the generation of only one token. Each station monitors the ring for invalid conditions requiring ring initialization. Invalid conditions include an extended

period of inactivity or incorrect activity. If an station detects that the time since it last saw a valid token significantly exceeds the Target Token Rotation Time (TTRT), then the station assumes an error condition. The error detection and correction involves the Claim Token Process, the Initialization Process, and the Beacon Process. Any station detecting the need for initialization of the ring initiates the claim process by issuing Claim frames. The MAC protocol uses this procedure to negotiate the same value for the Token Rotation Time (TRT) in all of the stations on the ring and to resolve contention among stations attempting to initialize the ring. The station that has won the claim process accomplishes the initialization process. The MAC protocol uses the beacon process to isolate a serious ring failure such as a break on the ring. [Ref. 13]

1. Facilities Specification

The facilities clause of the FDDI MAC and PHY standards define the means by which peer entities communicate on the ring. MAC facilities include symbol set, protocol data units formats, timers, and counts. PHY facilities are coding, symbol set, and line states. As background for protocol description and formal specification presented in this thesis it is relevant to describe the symbol set, formats of PDU, timers, and counts used by the MAC.

a. The FDDI Symbol Set

MAC and PHY operate similarly in a peer communication, however they use different signal units. MAC uses a *symbol* as an atomic signaling element to convey information; the PHY entity uses a code bit as the smallest signaling element. Code bits

are logical ones and zeros that represent optical signal polarity transitions by the use of NRZI encoding technique. A *symbol* is a group of five consecutive code bits. This sequence is also called as code group. Each code group provides 32 possible bit combinations. The establishment of code group boundaries is a concept implied in the definition of code group. This process is known as framing, and the established boundary is known as "framing boundary." Table 1 shows the FDDI symbol set mapped to code groups (adapted from the MAC standard). FDDI uses symbols to convey three types of information: **line state symbols**, **control sequences**, and **data quartets**.

TABLE 1: SYMBOL CODING (ADAPTED FROM MAC STANDARD)

Code Group		Symbol	
Decimal	Binary	Name	Assignment
		● Line State Symbols	
00	00000	Q	Quiet
04	00100	H	Halt
31	11111	I	Idle
		● Control Sequences	
		(a) Control Symbols:	
24	11000	T	First symbol of JK pair
17	10001	K	Second symbol of JK pair
13	01101	T	Ending Delimiter symbol
		(b) Control Indicators:	
07	00111	R	Reset (logical ZERO or OFF)
25	11001	S	Set (logical ONE or ON)
		● Data Quartets	*Hexadecimal* *binary*
30	11110	0	0 0000
09	01001	1	1 0001
20	10100	2	2 0010
21	10101	3	3 0011
10	01010	4	4 0100
11	01011	5	5 0101
14	01110	6	6 0110
15	01111	7	7 0111
18	10010	8	8 1000
19	10011	9	9 1001
22	10110	A	A 1010
23	10111	B	B 1011
26	11010	C	C 1100
27	11011	D	D 1101
28	11100	E	E 1110
29	11101	F	F 1111
		● Invalid Code Points	
01	00001	V or H	These code points shall not be transmitted because they violate run length or duty cycle requirements. Stream of codes points 01, 02, 08 and 16 shall be interpreted as Halt when detected.
02	00010	V or H	
03	00011	V	
06	00110	V	
08	01000	V or H	
12	01100	V	
16	10000	V or H	

There are three line state symbols: Quiet (Q), Halt (H), and Idle (I). These symbols are for use on the medium between transmission of Data Link Layer (DDL) protocol data units. The meaning of each line state symbols is as follows:

- Q indicates absence of activity on the medium.
- H indicates a logical break in activity on the medium.
- I indicates normal condition of the medium.

Control sequences are either control symbols or control indicator sequences. Control symbols are used to form the Starting Delimiter (SD) and Ending Delimiter (ED) sequences of a Protocol Data Unit (PDU). Control indicators specify logical conditions associated with a data transmission sequence (i.e., a MAC PDU). Control symbols are named J, K, and T. Control indicators are named R and S. The Encode function of PHY uses the symbol sequence "JK" from MAC to indicate the starting boundary of a PDU. This starting boundary is called the Starting Delimiter (SD) of a PDU. This symbol pair forms a uniquely recognizable group of code bits. The symbol "T" is the ending delimiter symbol used to terminate all PDUs. This control symbol shall appear in the Ending Delimiter (ED) field of a PDU. The ED field may use either one or two T symbols; if a PDU is as **frame** then the ED field contains only one T symbol. In this case the T symbol shall be followed by the Frame Status field that has a minimum of three control indicator symbols (R, S) to form a sequence with even number of symbols also called balanced sequence of symbol pairs. If a PDU is a **token**

then no control indicators are present; in this case the ED field contains two T symbols. As opposed with Starting Limit that has a uniquely recognizable code bit sequence regardless of previously established framing boundaries, the Ending Delimiter cannot be recognized as independent of symbol boundaries. Therefore, previous establishment of frame boundaries is necessary for proper decoding of this symbol.

A data quartet symbol conveys four data bits of arbitrary data within a frame. The hexadecimal digits (0-F) denote the sixteen data quartet symbols. The character "n" denotes a generic element of this set. The encoding technique used by PHY is called as 4B/5B since each four bits of data are encoded into a symbol of five cells, each cell contains a single signal element.

A violation symbol V denotes a condition on the medium that does not conform to any other symbol in the symbol set. Invalid code points are formed by V or H symbols. These Code Points are not to be transmitted since a violation on code run length and Direct Current balance requirements will occur.

b. Formats of Protocol Data Units

FDDI MAC specifies two formats of PDU: frame and token formats. Frame formats are variable-length PDU used for transmission of Data Link Layer messages. FDDI MAC controls the sizes of frames as required by the physical layer. The maximum frame length is 4500 octets or 9000 symbols. Tokens are short fixed-length PDU that allow the right to transmit data. Frames and tokens are structured in predefined sequences of fields. Each field contains one or more symbols ordered so that the left-

most symbol is to be transmitted first, and is the most significant bit. Figure 4 depicts the frame format.

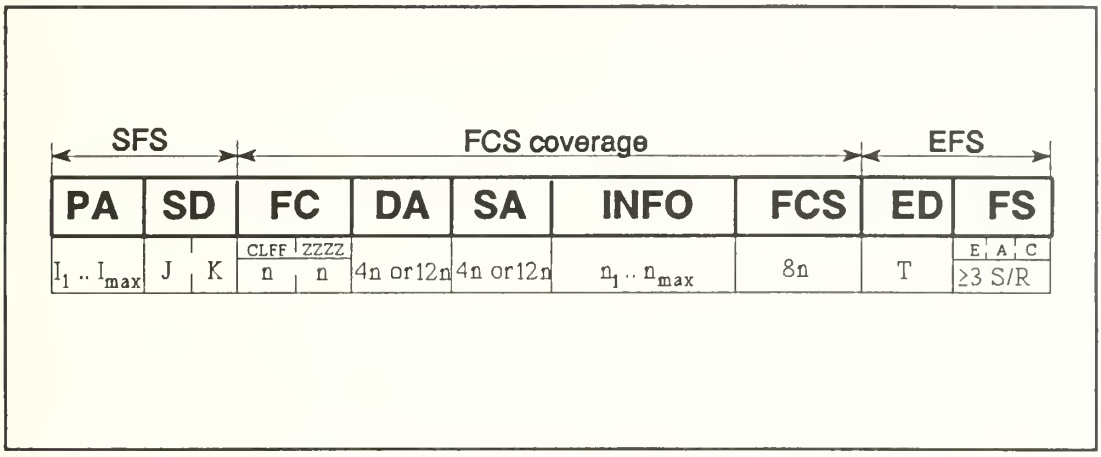


Figure 4: The Frame Format

Each field of the frame format has the following meaning:

- Preamble (PA) - consists of 16 or more Idle symbols to signal a start transition for synchronization of station's clock.

- Starting Delimiter (SD) - consists of two symbols (J and K) to signal a start receive of a frame.

- Frame Control (FC) - consists of two data symbols. These two symbols has the following eight bit format: CLFF ZZZZ. These bits indicate the Class (C) of service, the Length (L) of both MAC addresses (DA and SA), and the frame type (FF in conjunction with the CL and ZZZZ bits).

- Destination Address(DA) - consists of four or 12 symbols to indicate the destination address of the PDU.

- Source Address (SA) - consists of four or 12 symbols to indicate the originator of the PDU.

- Information (INFO) - consists of zero, one, or more data symbol pairs. These symbols forms the contents of the LLC, SMT, or MAC message carried by the frame.

- Frame Check Sequence (FCS) - consists of eight data symbols. This field is used to detect errors on data bits within the frame as well as erroneous addition or deletion of bits to the frame.

- Ending Delimiter (ED). - consists of one terminate symbol (T) to indicate a frame ending. The field is necessary to provide a criteria for acceptance of a valid frame. The ED must be met before a frame is accepted.

- Frame Status (FS) - consists of three or more Control Indicators symbols (R and S) that follows the Ending Delimiter of a frame. The first three Control Indicators are mandatory. They indicate Error Detected (E), Address Recognized (A), and Frame Copied (C).

c. Timers and Counts

FDDI is essentially a timed token rotation protocol. The MAC standard specifies a set of timers and counts to regulate and monitor ring operation. Each station maintains three timers to perform the timing requirements for the services: the Token Rotation Timer (TRT), the Token Holding Timer (THT), and the Valid-Transmission Timer (TVX). In addition, each MAC maintains frame counts as an aid to monitor the network performance, problem determination and fault location. Implementations may optionally employ other count, however three counts are mandatory: Frame_ct, Error_ct, and Lost_Ct. The next paragraphs briefly describe these timers and counts.

The purpose of TRT is to control ring scheduling during normal operation and to detect and recover from serious ring errors situations. TRT measures the time since a station last received a token in a rotation from one cycle to the next so that it defines if the token is "early" or "late." During different phases of ring operation the protocol initializes TRT with different values whenever it expires. The number of

TRT expirations is important information to assist Station Management in the isolation of serious ring errors. A counter called Late_Ct accumulates the TRT expirations.

The Token Holding Timer (THT) saves a time value for a dynamic bandwidth sharing or asynchronous service. This timer is initialized with the current value of TRT when a station captures the token. During the asynchronous service, THT is running to control transmission of asynchronous frames. A MAC may initiate a transmission of these frames if timer THT has not expired. In addition, a station shall release the token before its allocated THT expires.

The Valid-Transmission Timer (TVX) assists Station Management to recover from transient ring error situations. The MAC standard describes the derivations for a timeout TVX value called TVX_value. Once TVX expires it remains in this condition until reset by the Receiver.

Frame_Ct is the count of all frames received. This count is incremented whenever the terminate symbol (T) of a frame Ending Delimiter (ED) field is received.

Error_Ct is the count of error frames. This count is incremented if this MAC detects a frame error that no previous MAC has detected. This condition holds true when the Receiver sets an error flag for the arrived frame received with the Frame Status field showing the Error Detected indicator not set ($E \neq S$); otherwise, no error frame is counted by this MAC since the error have been already counted by other MAC.

Lost_Ct is the count of all instances in which MAC is in the process of receiving a PDU and an error is detected that prevents PDU reception. In these cases, MAC increments Lost_Ct and strips the rest of the PDU from the ring, transmitting idle

symbols. When remnants of PDU are received Lost_Ct is not incremented because they are followed by Idle symbols. The specification presented in this thesis shows precisely these instances for each incoming sequences that form the PDU. For example, whenever a format error occurs on the Starting Delimiter of a PDU the Receiver sends a FO_Error to the Transmitter, increments the Lost_Ct, and enters the AWAIT SD state for a new PDU. Then, the Transmitter begins to transmit Idle symbols stripping the PDU from the ring. The specification covers similar operation for all PDU field sequences which comes with a format error or whenever a PH_invalid signal is received from PHY.

2. Operation

This subsection briefly discusses several characteristics of the protocol operation, which are of interest for the formal specification. The timed-token mechanism, ring scheduling, frame and token transmission, and frame stripping are discussed.

a. FDDI Timed-Token Access Method

By passing a token around the ring, FDDI controls the opportunity that each station will have to transmit a frame or a sequence of frames. A token is a specific bit sequence that circulates among the nodes on the ring, giving transmit permission to any station that wants to transmit its data. Once a station "captures" a token, its frames may be transmitted. However, the access to the network and scheduling is also controlled by timers. The next paragraphs provides the details of the timing rules for the ring scheduling process of a FDDI network.

A key parameter set by the managers of FDDI networks is the Target Token Rotation Time (TTRT) [Ref. 2]. This is a value negotiated between all stations MAC during ring initialization via the Claim Token process. As part of the Claim Token process at the time of ring initialization, each MAC station uses a requested TTRT value (T_Req) to negotiate for the lowest operative value of TTRT (T_Opr). This value is required to be in the range of Token Rotation Timer (TRT), which is established for all stations as the closed interval from a minimum to a maximum TTRT value to be requested (T_Min and T_Max). As a result of this negotiation, the lowest value of T_Req becomes the negotiated TTRT value (T_Neg) at the Receiver of each station. The MAC winning Claim Token station then sets the operative TTRT value (T_Opr) to the negotiated TTRT value (T_Neg).

Tokens may be "early" or "late." A token which arrives before TRT reaches TTRT is an "early" token, otherwise is a "late" token. TRT is reset to T_Opr each time an early token arrives. An early token may be used for both classes of services synchronous or asynchronous, whereas a late token may be used only for synchronous service.

The FDDI protocol guarantees an average TRT not greater than TTRT, and a maximum TRT not greater than twice TTRT. Johnson in [Ref. 3] proved that the timing requirements of this protocol are satisfied.

Figure 5 illustrates how T_Opr for the ring is obtained during ring initialization. This figure shows a timing chart for n FDDI stations. The station number two is the winning Claim Token station since its T_Req is the lowest of all requested

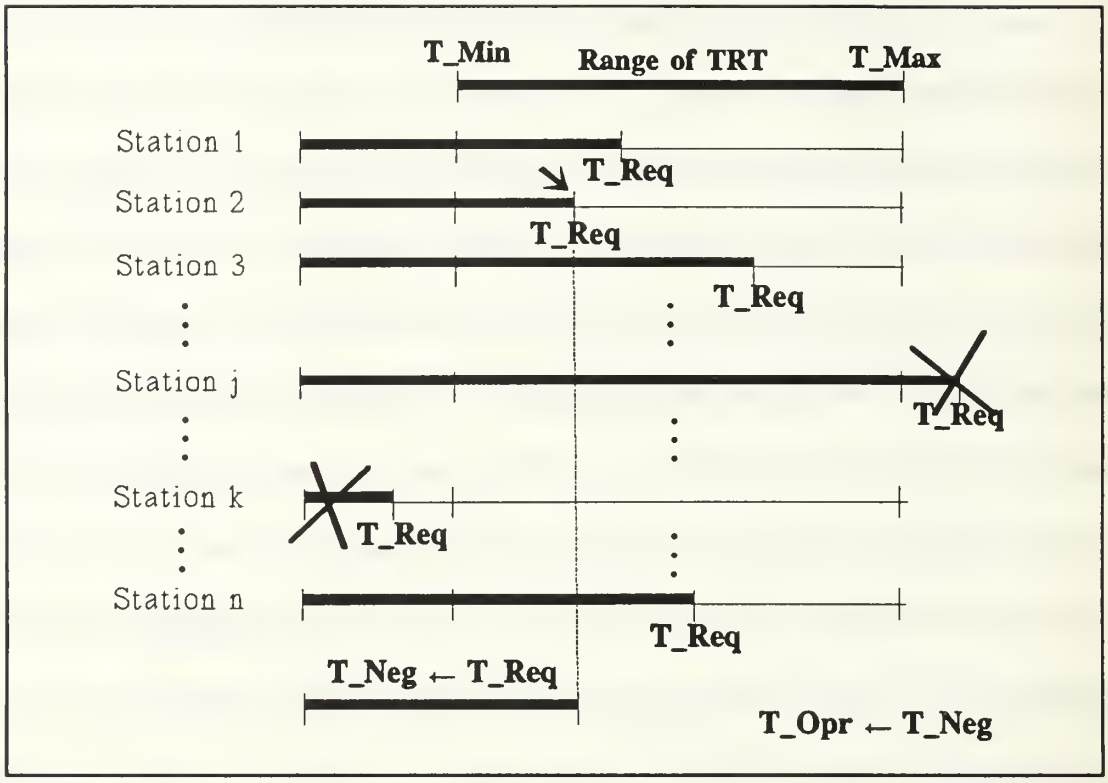


Figure 5: Derivation of T_Opr During Ring Initialization

TTRT values that fall in the range of TRT. Note that if a T_Opr falls outside the range of TRT then a station is unable to operate correctly on the ring (stations J and K).

The Token Rotation Timer (TRT), the Token Holding Timer, the station's synchronous bandwidth allocation, and the counter Late_Ct govern the amount of time that a station may hold the token and transmit frames [Ref. 4]. The timed-token rules of FDDI are summarized as follows:

- If Late_Ct = 0 (token early), then the Transmitter places the current value of TRT into THT, and resets TRT to T_Opr. This is represented in the formal specification as THT ← TRT; and TRT ← T_Opr; both synchronous and asynchronous frames may be transmitted.

- If Late_Ct > 0 (token late), the value "expired" is placed into THT, and Late_Ct is cleared (THT ← expired; Late_Ct ← 0;). In this case, TRT is not reset to T_Opr and only synchronous frames may be transmitted.

- During synchronous transmissions only TRT is running. During asynchronous transmissions both TRT and THT are running.

- No frames are allowed to be transmitted after expiration of the station's TRT. The length of time an individual station may transmit synchronous frames is bounded above by its synchronous bandwidth allocation. The THT limits the time for asynchronous frames.

In the formal specification the station MAC Transmitter is responsible to carry out the timing operations. The model allows the representation of ring scheduling in the FDDI network. The MAC Transmitter State Diagram and the Transition Table show the representation of these rules.

b. Frame and token transmission

Upon a request for Service Data Unit (SDU) transmission, MAC constructs the Protocol Data Unit (PDU) or frame from the SDU by placing the SDU in the INFO field of the frame. The SDU remains queued by the requested entity awaiting for the receipt of a token. After the token is captured the station's MAC transmits its queued frames according to the rules of the token holding. [Ref. 19]

After the token holding station completes the transmission of frame or frames, the MAC immediately issues a new token. The standard leaves as optional the implementation of a MAC which may wait to see one or more frames return before issuing the token.

c. Stripping

The stripping method of FDDI defines the frame originator as the station responsible for frame removal from the ring. Since the decision to strip a frame is normally based upon the recognition of the MAC's address in the SA field, which cannot occur until after the initial part of the frame has already been repeated, some remnants of frames continue to circulate on the ring. These remnants consists most of the PA, SD, FC, DA, SA, and six symbols after the SA field, followed by idle symbols. This truncated frame will not cause problems to the ring because all other stations will recognized these sequences of symbols as a remnant since they are followed by idle symbols and no terminate symbol "T" will be received. With the formal specification presented in this thesis the MAC Receiver establishes a check for remnants in every field of the incoming PDU, which enhances the protocol error checking specification. Also, Chapter III will show that the stripping method in the normal operation of the improved protocol is changed such that the Destination Address (DA) station is responsible for the removal of Logical Link Control (LLC) or Station Management (SMT) frames from the ring. This leaves less remnants fields of frames on the ring since the DA comes first in a PDU sequence of fields.

3. Service Specification

The service specification defines a set of functions that one layer or sublayer entity provides to its users above or to management entities. These functions are defined in terms of primitives and parameters. The primitives describe the operations carried

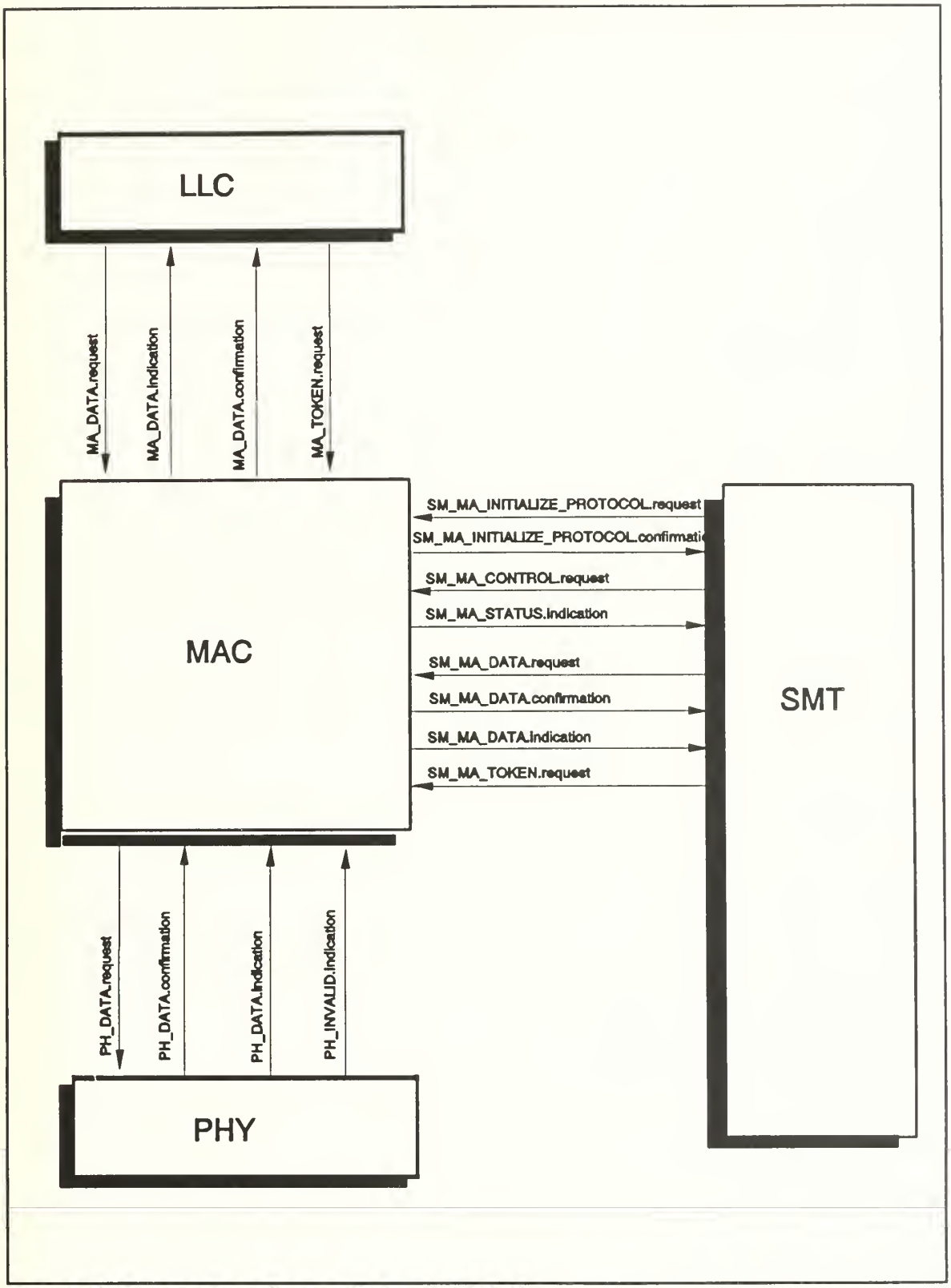


Figure 6: Service Specification Related to MAC

through the interface in which the service is provided. The parameters are associated with the primitives and the execution of a service primitive depends on the exchange of parameters between two entities. The services execution of the protocol observes a strict hierarchy on the sequence of operations. At any given moment the allowed primitive and parameter values depend on preceding history of operations. The model for protocol specification used in this thesis is suitable to specify the FDDI protocol because it represents precisely this sequence of operations. Although the standards specify clauses with mandatory services they also specify optional services. Furthermore, the standards accept as equally valid any implementation technique that causes the same external behavior of the protocol. Figure 6 shows the set of services that MAC supplies to the local LLC entity and SMT entity and also the services required by MAC from the local PHY entity. The next paragraphs describe the contents of some primitives to illustrate the service specification provided and required by FDDI MAC standard.

By using a set of fixed-length symbols, peer MAC entities communicate on the ring. All protocol data units generated by peer Data Link Layer entities are matched pairs of symbols. However, each symbol is sent across the Physical layer sequentially. On the transmitting station, MAC conveys information to local PHY by a continuous sequence of symbols via the defined primitives. In a MAC-PHY transfer of data, a PH_DATA.request primitive is used and the parameter PH_Request(symbol) is sent to PHY whenever MAC has a symbol to output as shown in Figure 7. Upon receipt of this primitive the PHY entity performs the encoding and transmits the symbol. For every PH_Request received from MAC, PHY returns a PH_confirmation to provide a

synchronization of the MAC data output with the data rate of the medium and to indicate its readiness to accept another symbol. On the receiving station, a PH_DATA.indication primitive is used to define the transfer of data from PHY to MAC. This indication occurs whenever PHY decodes a symbol. Also, the receiving MAC shall only recognize the incoming PDU as matched pairs of symbols.

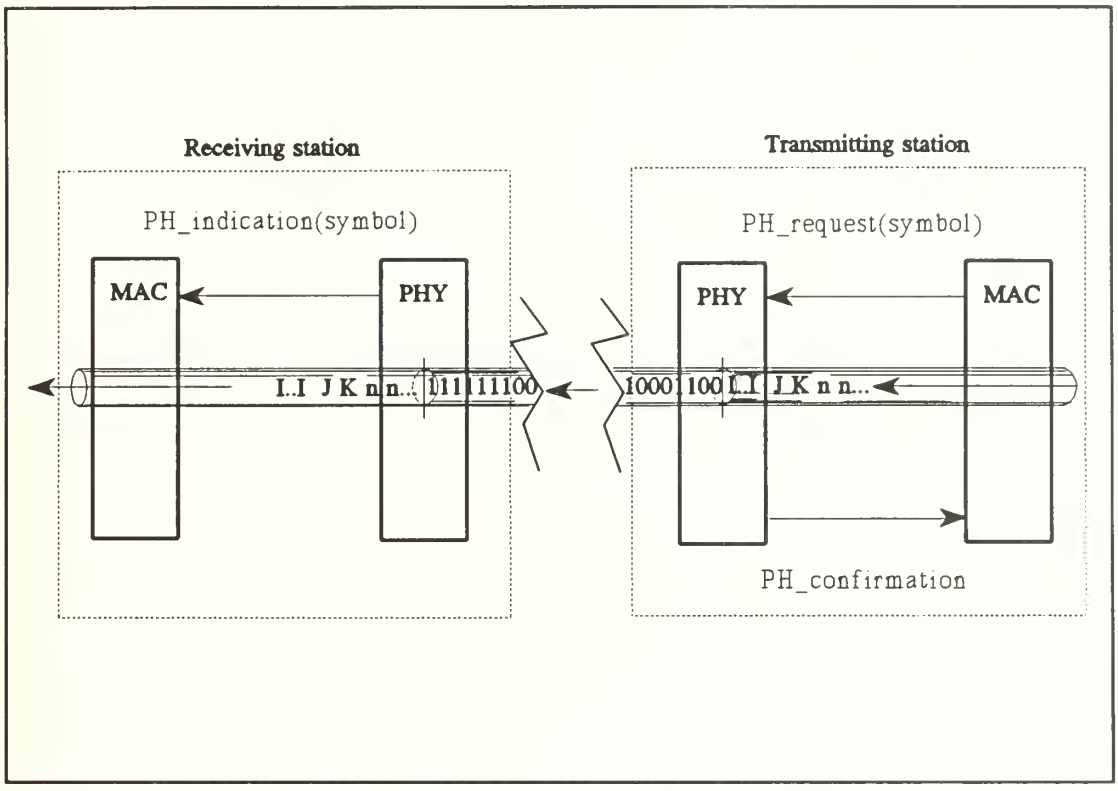
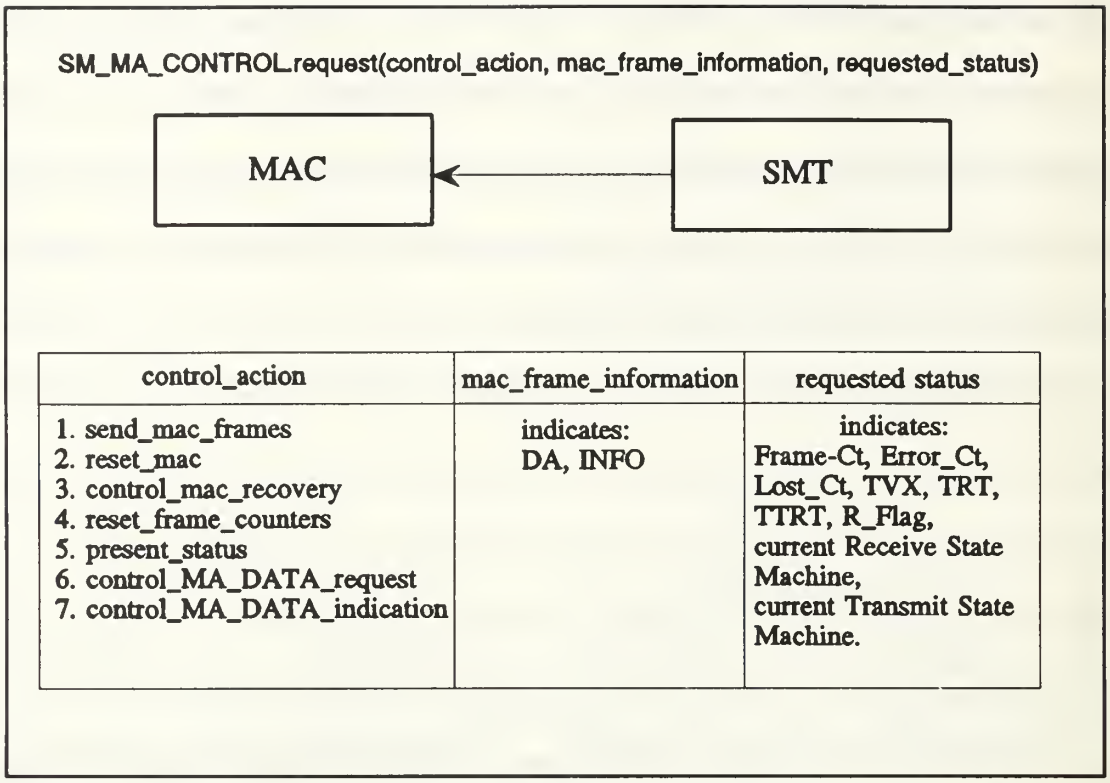


Figure 7: The MAC-PHY Interface Service Primitives. MAC passes symbols to PHY, which translates them into bits and then transmits the bits as optical signals.

SM_MA_CONTROL.request is a primitive used by SMT to control the operation of MAC. Figure 8 illustrates this service primitive at the interface MAC/SMT with a table that contains the parameters and its associated values.



control_action	mac_frame_information	requested status
1. send_mac_frames 2. reset_mac 3. control_mac_recovery 4. reset_frame_counters 5. present_status 6. control_MA_DATA_request 7. control_MA_DATA_indication	indicates: DA, INFO	indicates: Frame-Ct, Error_Ct, Lost_Ct, TVX, TRT, TTRT, R_Flag, current Receive State Machine, current Transmit State Machine.

Figure 8: The SM_MA_CONTROL.request service primitive

SMT generates this primitive to cause MAC to take the specified control action. For example, if the control action is reset_mac, then MAC generates the MAC_Reset signal. This signal will be an enabling predicate for transitions to occur in both MAC Receiver and Transmitter. The formal specification presented in this thesis will show these transitions in the state diagrams and will describe them in the transition tables.

As a response to SM_MA_CONTROL.request(control_action = present_status) MAC provides the SM_MA_STATUS.indication service primitive to SMT. This primitive contains a status_report parameter, shown in Figure 9. The FDDI standard specifies some of the status_report parameters as optional. The work presented

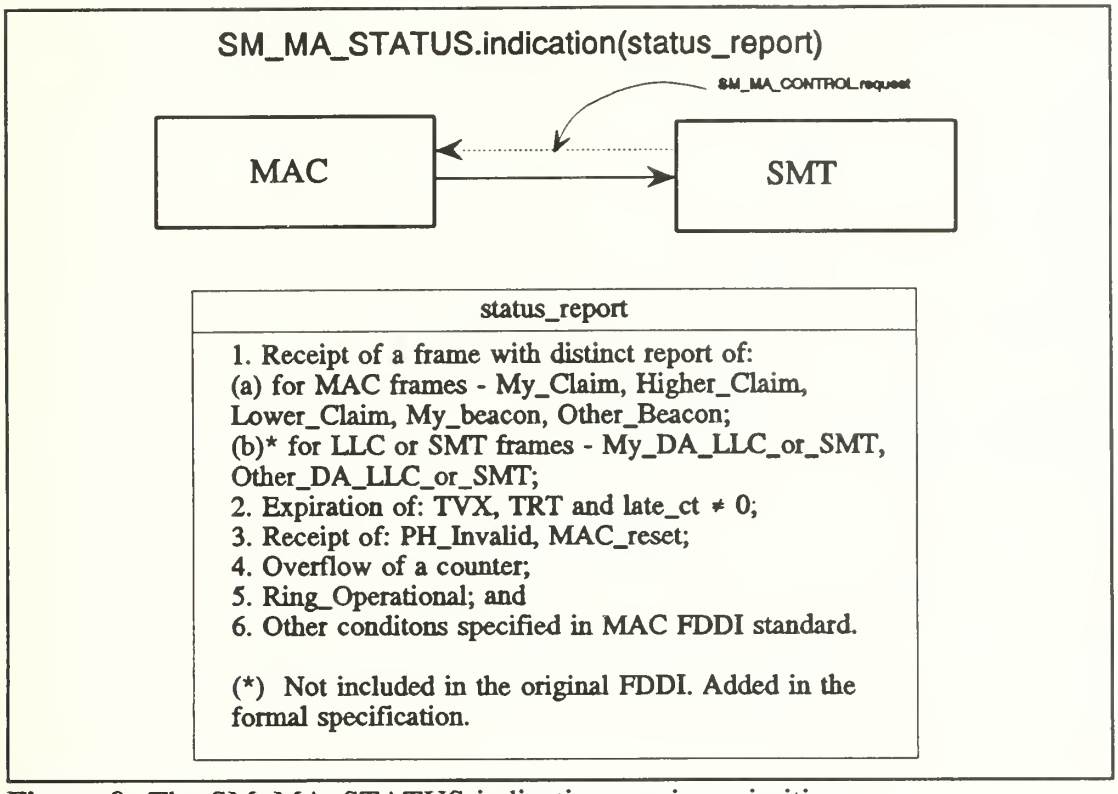

SM_MA_STATUS.indication(status_report)
SM_MA_CONTROL.request
MAC
SMT
status_report
1. Receipt of a frame with distinct report of:
(a) for MAC frames - My_Claim, Higher_Claim, Lower_Claim, My_beacon, Other_Beacon;
(b)* for LLC or SMT frames - My_DA_LLC_or_SMT, Other_DA_LLC_or_SMT;
2. Expiration of: TVX, TRT and late_ct ≠ 0;
3. Receipt of: PH_Invalid, MAC_reset;
4. Overflow of a counter;
5. Ring_Operational; and
6. Other conditons specified in MAC FDDI standard.
(*) Not included in the original FDDI. Added in the formal specification.

Figure 9: The SM_MA_STATUS.indication service primitive

in the Chapter IV of this thesis requires a more detailed analysis of the actions taken by the MAC state machines. Additional variables are included in the formal specification.

C. DESCRIPTION OF THE EXISTING FORMAL MODEL "SYSTEMS OF COMMUNICATING MACHINES"

This section describes the model *systems of communicating machines*. This model was designed as a method for the formal description and verification of communication protocols [Ref. 8]. It has been used in the specification of local area network protocols such as CSMA/CD and token ring [Ref. 10]. The model description given below is found in [Ref. 9].

A *system of communicating machines* is an ordered pair $C = (M,V)$, where

$$M = \{m_1, m_2, \ldots, m_n\}$$

is a finite set of *machines*, and

$$V = \{v_1, v_2, \ldots, v_k\}$$

is a finite set of *shared variables*, with two designated subsets R_i and W_i specified for each machine m_i. The subset R_i of V is called the set of *read access variables* for machine m_i, and the subset W_i the set of write access variables for m_i. The integers n and k are the number of elements (machines and variables) in sets M and V.

Each machine $m_i \in M$ is defined by a tuple $(S_i, s_0, L_i, N_i, \tau_i)$, where

(1) S_i is a finite set of states;
(2) $s_0 \in S_i$ is a designated state called the *initial state* of m_i;
(3) L_i is a finite set of *local variables*;
(4) N_i is a finite set of transitions names. Associated with each name is a unique triple (p,a), where p is an *enabling predicate* on the variables of $L_i \cup R_i$, and a is an *action* on the variables of $L_i \cup R_i \cup W_i$. Specifically, an action is a partial function

$$a: L_i \times R_i \rightarrow L_i \times W_i$$

from the values contained in the local variables and read access variables to the values of the local variables and write access variables.
(5) $\tau_i : S_i \times N_i \rightarrow S_i$ is a transition function , which is a partial function from the states and names of m_i to the states of m_i.

Machines model the entities, which in a protocol system are processes and channels. The shared variables are the means of communication between the machines. Intuitively, R_i and W_i are the subsets of V to which m_i read and write access, respectively. A machine is allowed to make a transition from one state to another when the predicate associated with the name for that transition is true. Upon taking the transition, the action associated with that name is executed. The actions changes the values of local and/or shared variables, thus allowing other predicates to become true.

Let $\tau(s_1, n) = s_2$ be a transition which is defined on machine m_i. (That is, τ is the edge pointing from state s_1 to state s_2). Transition τ is *enabled* if the enabling predicate p, associated with the name n, is true. transition τ may be executed whenever m_i is in state s_1 and the predicate p is true (enabled). The execution of τ is an atomic action, in which both the state change an the action a associated with n occur simultaneously.

The sets of local and shared variables specify a name and a range for each. In most cases, the range will be a finite or countable set of values. For proper operation, the initial values of some or all of the variables should be specified.

D. PREVIOUS WORK ON IMPROVEMENT OF FDDI

1. Suggestions on Improvement of FDDI

This thesis continues the work on improvement of FDDI documented in [Ref. 11]. In that paper, Lundy makes the observations of little use of the secondary ring and excessive frame propagation on the primary ring and he presents alternative transmission procedures that can lead to a network throughput of three to four times of the standard. This thesis takes the suggested procedures for transmission of frames that raises the ideal upper bound to 300 Megabits per second to develop the improved protocol and provide its formal specification. This subsection will review this previous work with additional comments.

The suggested alternative transmission procedure attempts to increase parallelism, decrease unnecessary frame propagation, and make maximum utilization of all available fiber. The basic Timed-Token access method of FDDI is not changed. The goal is to maximize the utilization of both rings by the Token Holding Station (THS), and allow other stations to use an available segment of the ring concurrently with the THS. The next paragraphs describe this procedure with illustrations.

Two main changes are proposed to improve the throughput in the FDDI protocol: concurrent access, and simultaneous transmission on the dual ding. The first change allows an additional frame to be transmitted in parallel on the same ring with the

single frame allowed by the standard token ring protocol. The second change allows a station in possession of the token to transmit on both rings, in opposite directions, simultaneously. The first procedure is suggested as follows:

- Frames transmitted from station *i* to station *j* are removed from the ring by station *J* rather than propagating around the ring. The frame is sent on the ring 1, which has the shortest distance from *i* to *J* (Figure 10).

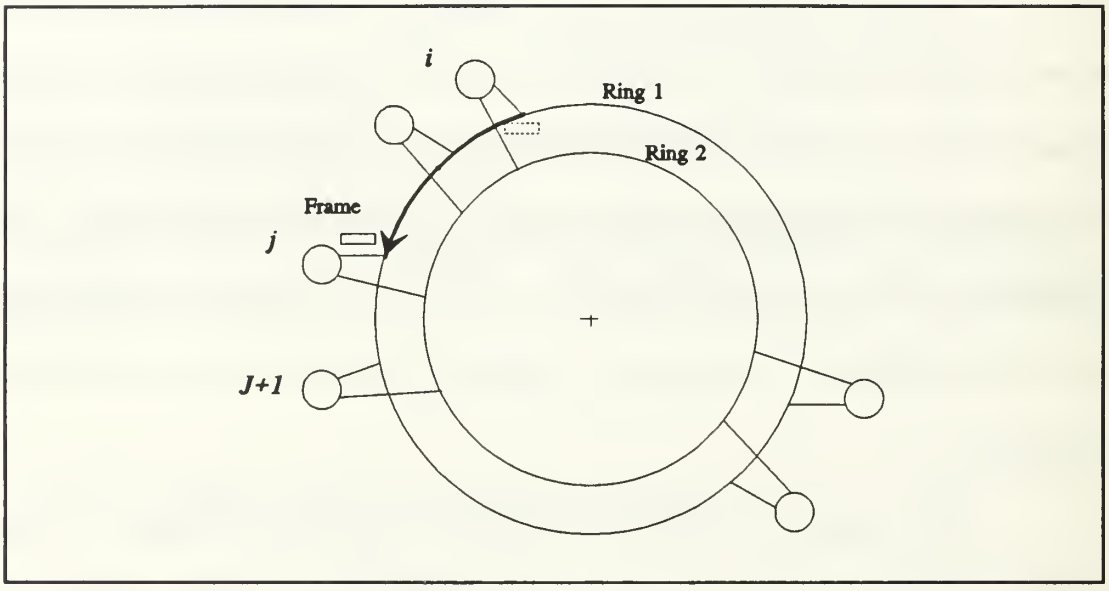


Figure 10: Step 1: frame transmitted from station *i* to station *j* and removed by *J*

- The acknowledgment from *i* to *j* is sent as a short message on the opposite ring, (ring 2) also on the shortest path (Figure 11).

- The remainder of the ring may now be used for transmission of another message in parallel with *i*'s message. Station *i* passes a "subtoken" to the next station *after* station *j*, which specifies the first and the last stations on the "open" segment of the ring. The subtoken is included as part of header of the frame sent to station *j*, which transmits it (the subtoken field only) on to station *j+1*.

The subtoken gives the right to only one station to transmit on the unused portion of the ring. If station *j+1* has no messages for any station on the segment, the subtoken is passed on to the next station. This is repeated until either one frame

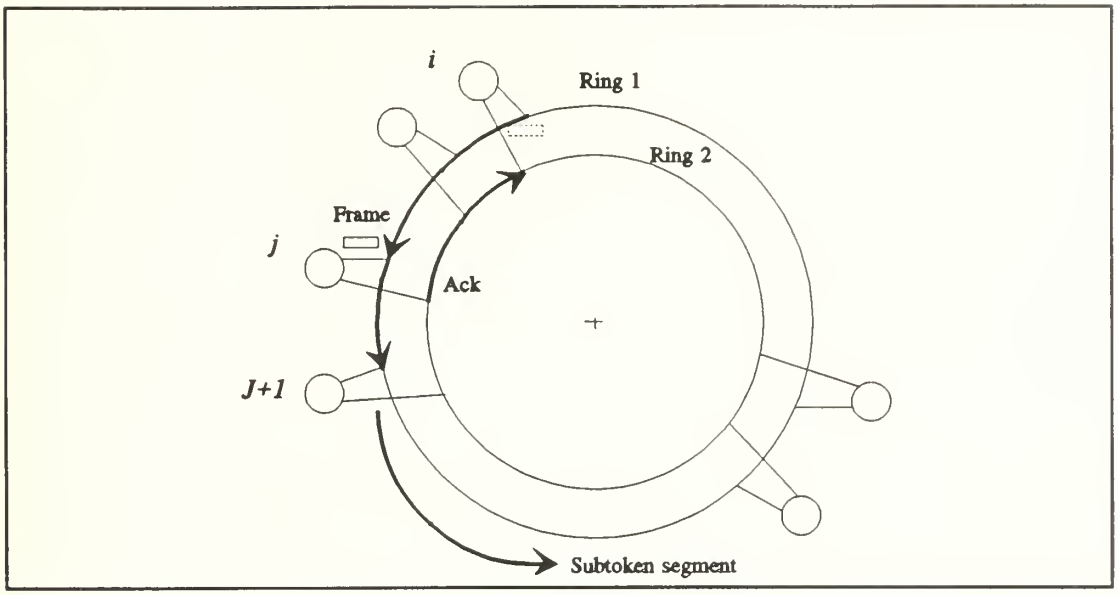


Figure 11: Step 2: the ack is sent back on the opposite ring and station *j* passes a "subtoken" downstream on the freed ring segment.

is transmitted on the segment, or the subtoken dies at the last station.

The length of the frame sent on the "subtoken segment" must be limited so that transmission will complete before station *i* finishes transmission to j and issues the next subtoken. The length should be at most the length of the frame sent by the station with the main token.

The second change from the transmitting protocol is to allow a station in possession of the token to transmit on both rings, in opposite directions, simultaneously (if it has more than one frame to transmit). This implies that each station must keep two queues for transmission, one for each ring. To avoid collisions between frames the ring is partitioned by the destination address station (Figure 12).

One of the main problems that occur with full dual ring operations is the issue of distributing the load onto the two available rings. Lundy analyzes two criteria for queuing up the frames in each ring. The shortest path and the load balanced among the queues.

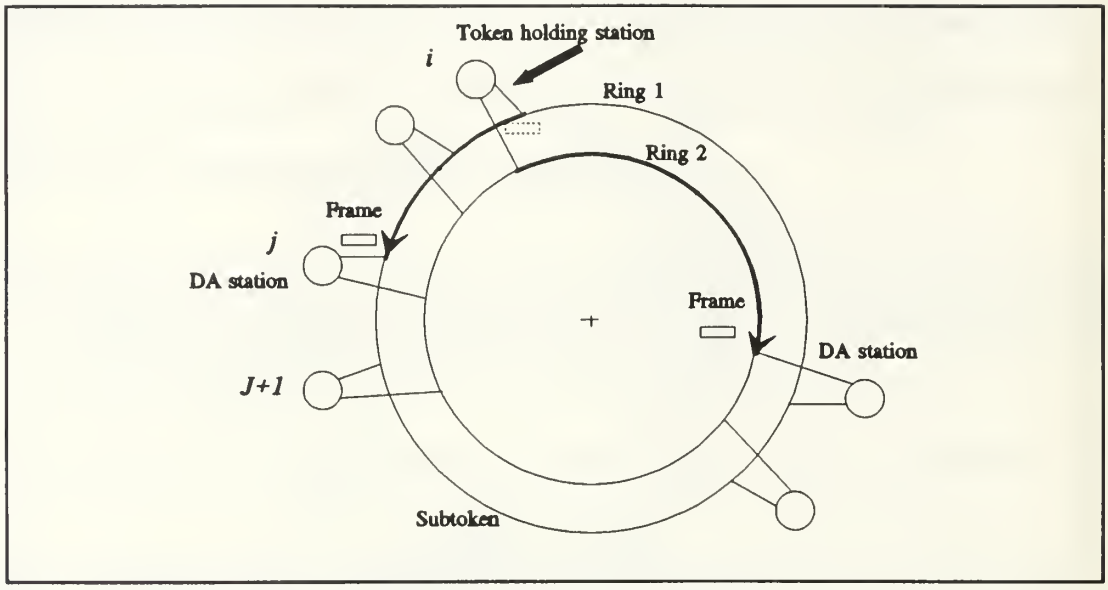


Figure 12:Simultaneous Transmission on the Dual Ring.

The shortest path criteria would avoid collisions, however it is possible that one queue would be longer than the other, so that parallelism and thus throughput would not be maximized.

The other criteria, which the author believes to be preferred, is to keep the two queues balanced. This is possible to do without collisions also; however it could become necessary to move frames from one queue to the other, to avoid collisions and keep the two queues in balance. [Ref. 11]

The criteria chosen to distribute the load in the protocol specified in this thesis is the load balanced queue. In addition to these previous observations, the shortest path is somewhat efficient for the simultaneous transmissions of the token holding station in an ideal situation of matched queued load for both halves of the ring; however, for the transmissions of the subtoken holding station the shortest path criteria is clearly undesirable. If a subtoken holding station wishes to send its frames on the subtoken segment which is greater than the one half of the total ring path then the shortest path criteria cannot be applied to this station. Furthermore, if different criteria for distributing the load in the token and subtoken holding stations is used, then the protocol overhead

may substantially be increased. Therefore, it is assumed a load balancing algorithm applied above the MAC layer to maximize the protocol efficiency.

This previous work forms the basis for the development of a more detailed protocol MAC structure presented in Chapter III and its formal specification in Chapter IV. The FDDI is a complex protocol and changes of this nature open several important questions that need to be carefully analyzed. One of the questions is how to guarantee the required reconfiguration capability of the network on a dual ring operation? Another question is how the initialization process will work in the proposed partitioned ring design. Also, it was stated that the dual ring of trees topology of FDDI offers the flexibility and availability to meet many requirements. With the improved protocol is it possible to maintain these similar features on a network? These and other questions are discussed in subsequent topics covered in the thesis.

2. Other Work on Improvement of FDDI

The increasing need for high speed data rates in Local Area Networks has lead to much work on improvement of token ring architectures including FDDI.

In [Ref. 1] Cidon and Ofek explain Metaring, a network that uses a full-duplex ring with spacial reuse and concurrent access to achieve higher throughput rates. This network employs two basic modes of operation: buffer insertion for variable sizes PDUs and a slotted ring for fixed PDU.

There are similarities and differences between Metaring and the improved protocol presented in this thesis. Table 2 presents an overall summary of points in common and contrasts between these two protocols.

TABLE 2: SIMILARITIES AND DIFFERENCES BETWEEN METARING AND THE IMPROVED FDDI PROTOCOL

Metaring	Improved FDDI
● To improve throughput	
Uses both rings for concurrent access and disjoint ring partitions for simultaneous transmissions. Four nodes transmitting at the same time in each direction on both rings. Maximum throughput: 800 Mbps	Uses both rings for concurrent access and disjoint ring partitions for simultaneous transmissions. Two nodes transmitting at the same time. The timed-token node transmitting in each direction on both rings; and the subtoken node transmitting in either one or other direction within its segment. Maximum throughput: 300 Mbps
● Distributing traffic across the two rings and stripping method	
(a) Shortest path; and (b) Frames removed by Destination Address.	(a) Load balancing algorithm; and (b) Frames removed by Destination Address in normal operation and removed by Source Address in "wrap condition."
● Fairness	
Use of single control message rotating in the opposite direction to the data traffic that it regulates.	Use the same mechanism of FDDI, the timed-token access method; which has been proven to be fair and deterministic.
● Access control to the physical medium	
Two access modes: (a) Buffer insertion for PDU of variable size; and (b) Slotted ring for fixed PDU.	Timed-token controlled concurrent access which employs: (a) One main Token for the two segments on both rings controlling simultaneous transmissions; and (b) One subtoken on the third segment of the ring controlled by the token holding station for transmissions within this segment; thus, this access method allows concurrent transmissions on disjoint segments of the ring.
● Ring scheduling and priorities	

Allows both classes of service: synchronous and asynchronous; and Allows priorities.	Allows both classes of service: synchronous and asynchronous; and Allows priorities.
● Addressing	
Arranged in an increasing order (1..n)	Same as in FDDI
● Problems of access control	
(a) May suffer from starvation - solutions are presented	The timed-token controlled concurrent access avoids the problem of starvation since the three segments of the ring used for concurrent transmissions are logical disjoint parts for ring access. There are only two transmitting nodes at a time: the token and the subtoken holding station. The token holding station controls the duration of the subtoken.
(b) Bandwidth reservation - problem of how to guarantee delay to some of bandwidth, while still allowing asynchronous distributed access. Solution discussed.	No bandwidth reservation problem. Uses the same mechanism for service assignment employed in FDDI. First, allocates synchronous bandwidth, then asynchronous.
(c) Priority - the distributed nature of the access is not allowing the implementation of a priority access scheme. Solution discussed.	No priority problem. Same priority scheme used in FDDI.
(d) Delay bounds - problem associated with buffer insertion architecture.	Same delay of FDDI.

One of the key characteristics of FDDI is its reliability feature embedded in the dual ring architecture. Although the improved protocol makes use of both rings during normal operation and uses the destination address to remove a frame, when a link or node fails the protocol switches the frame stripping mechanism and works as in the original FDDI using only one path. Therefore, this is one of the main advantages of this protocol. The improvement achieved in utilization does not degenerate the reliability function of FDDI.

In [Ref. 16] Strohl briefly discusses the variety of choices on using FDDI's dual ring. Among those choices, the protocol developed in this thesis fits in the category

of using the second ring for traffic in normal operation. However, by design decision, the MAC structure of this protocol differs from the current structures of MAC used in the original FDDI; consequently, different analysis can be done. Unlike the suggestion for the use of both rings employing dual MAC stations to potentially achieve a bandwidth of 200 Megabits per second, this thesis developed a protocol that uses a single MAC to control simultaneous access to both rings. This single MAC structure is presented in Chapter III. Chapter III will also show that with the use of a single MAC controlling the operation of both rings this protocol enhances the end-end connectivity through ring configuration changes.

Other work on improvement of token ring networks have appeared on the literature. In [Ref. 5] Kamal proposes the use of multiple tokens to circumvent the problem of excessive delay under a very light network load. Again, among other differences, this multiple token network differs from the protocol proposed in this thesis primarily in the medium access control method. In [Ref. 15] Siegel analyzes hardware and software functions for a dual ring operation in FDDI. An architecture for an enhanced FDDI station for traffic acceptance and distribution is presented.

III. THE IMPROVED FDDI PROTOCOL

This chapter describes the improved FDDI protocol. Section A introduces the improved FDDI MAC/PHY structure, the changes needed in the original MAC, and the new protocol data units formats. In section B, the mechanism of access control which enhances ring utilization is presented.

A. PROTOCOL DESCRIPTION

Although the protocol described in this thesis differs from the original FDDI in the utilization of the medium, many other characteristics of operation are the same in both protocols. This includes the timed-token access method. Stations wait for a passing token to transmit their data. They agree in a target time for a token rotation. The timers TRT and THT and the counters work in the same fashion. The processes for claiming a token, ring initialization, and beacon remain unchanged. The Ring scheduling for the improved protocol supports both classes of service as in the original FDDI. In fact, the key difference between the two protocols is the use of a second access control Protocol Data Unit, called "subtoken", which gives the right to another station to transmit concurrently with the timed-token station, potentially improving ring utilization. However, the employment of this special PDU in the improved protocol is controlled by the MAC station that is holding the main token; the parallel access to the medium granted by the subtoken does not interfere with the access given by the main token. In addition, this

protocol also uses the second ring to carry out simultaneous transmission increasing the network throughput to a maximum of 300 Megabits per second.

1. The Improved FDDI MAC/PHY Organization

To achieve the high performance with simultaneous use of both rings this protocol uses an specific structure for the trunk ring MAC/PHY local entities. Figure 13 depicts this structure. This figure shows a single MAC to control services provided from both physical layers. This structure requires a Dual Attachment Station (DAS) with a single MAC. The high throughput achievement is essentially based on the use of the dual trunk ring; therefore, slave stations which are single attached to concentrators will take advantage of the improvement, however they do not contribute to this process. The proposed solution for the problem of Single Attached Stations (SAS) with the simultaneous use of both rings in this protocol is to provide a Dual Attached Concentrator (DAC) with an enhanced MAC capability. This MAC has the same structure of all MACs of Dual Attached Stations on the trunk ring with an additional capability to respond for each of its slaves as if each slave is directly attached to the dual trunk ring.

The single MAC organization adds a degree of complexity; however, the benefits overcome this cost. Synchronization of both rings in a data transfer at 100 Megabits per second is a challenging problem to solve in an FDDI implementation that uses dual MAC; however, with a single MAC controlling both ring transmissions this problem seems to have a feasible solution. In FDDI, a station MAC placed in one ring is not allowed to communicate with a MAC placed in another ring. In the protocol

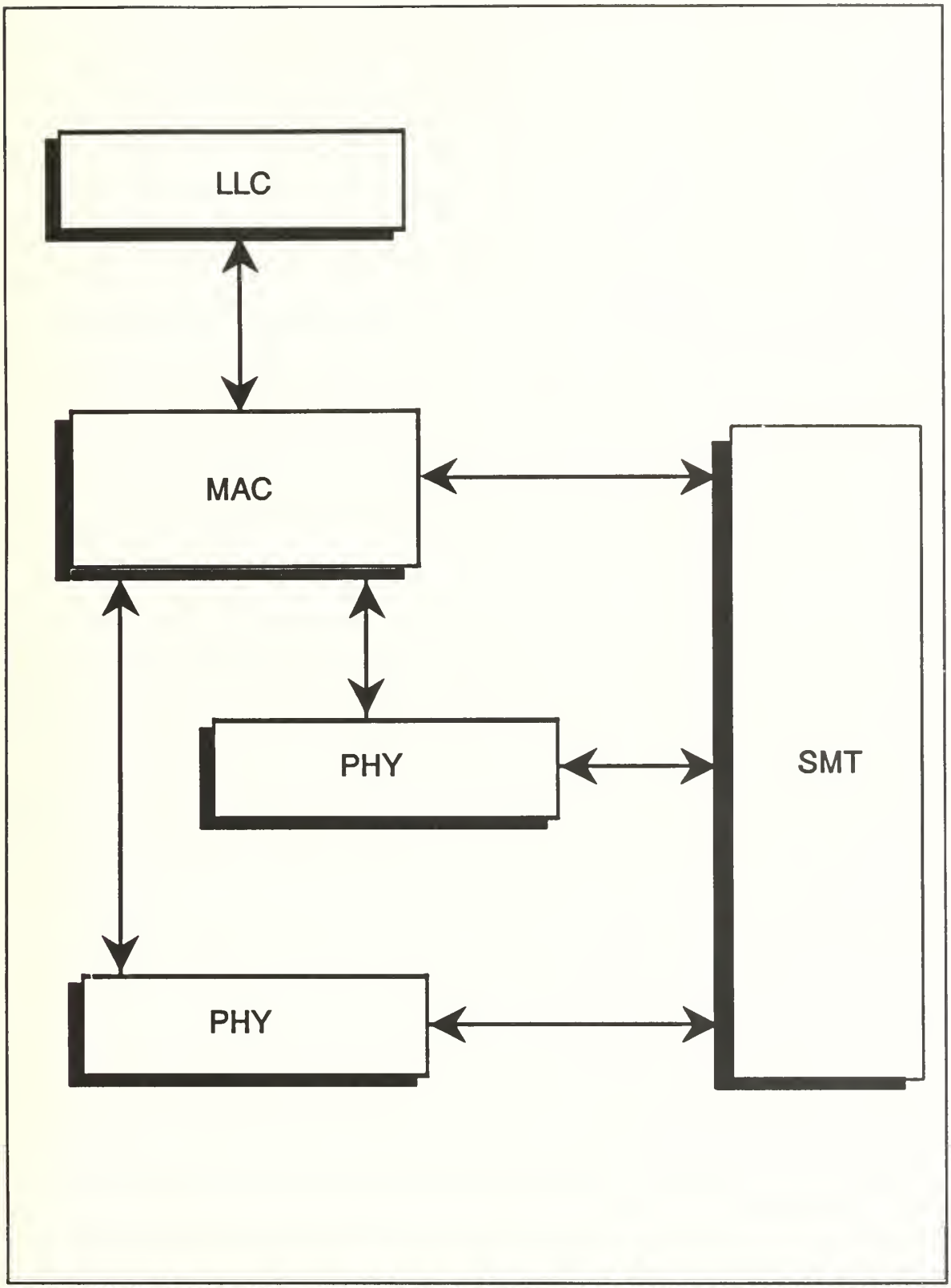


Figure 13: Structure of Local Entities in the Improved FDDI (Single-MAC-Dual-PHY).

developed in this thesis, the solution proposed for a dual ring operation is a Single-MAC-Dual-PHY structure. This unique structure has to be more complex than the current structures specified by FDDI. Due to its complexity, this Single-MAC-dual-PHY structure can be a subject for more work conducting an in depth analysis.

One advantage of a single MAC to control the operations of both rings is the enhancement of end-end connectivity through ring configuration changes. In [Ref. 16] Strohl analyzes two troublesome cases for a dual MAC station to ensure connectivity if both rings are used for data transmission. The problem deals with determination of the correct MAC to maintain communications through configuration changes (WRAP to THRU and THRU to WRAP). The Two cases are:

- The ring is in a WRAP condition and two stations establish a connection. Then, the configuration changes to THRU. The connected MAC can no longer communicate since they are on different rings. The problem is determination of MAC placements of a dual MAC wrapped station when the configuration changes to THRU in order to choose the correct MAC to maintain communications.
- The ring is in a THRU condition and two stations establish a connection. Then, one of these stations wraps with one of the connected MAC off of the wrapped ring. This disrupts the communication path.

With a Single-MAC-Dual-PHY structure these problems no longer exist since if a configuration change occurs on the network this MAC continues in the path to allow stations connection. Therefore, this MAC/PHY structure enhances end-end connectivity through configuration changes. Connectivity is never lost with a single MAC controlling access to both rings.

2. Changes in the Original Protocol

To allow concurrent access to the physical medium and simultaneous use of both rings two basic changes are implemented in the original FDDI. The first change is the establishment of dynamic logical partitions on both rings. As a result of these partitions, a change in the method of striping the frames from the rings is necessary. Three logical segments divide the physical medium in the improved protocol. Figure 14 illustrates these partitions. Three stations on the ring form the vertices that join the three segments. One station is the token holding station (THS), which captured the token. There will be only one token circulating on this dual ring network. The token holding station establishes the ring partitions by transmitting frames simultaneously on both rings in opposite directions. The Destination Addresses (DA) contained in each of the two frames establish the two other vertices on the partitioned rings. The segment of the rings not covered by transmissions of the token holding station can be used by a second station to transmit its data concurrently on the ring. Therefore, two stations can access different parts of the medium at the same time.

Concurrent use of the physical medium requires changes in the method of frame removal from the ring and acknowledgments of frame received. In FDDI, all frames make a complete rotation on the ring and are then removed by the their originators. The advantage of this procedure is simplicity. The disadvantage is the innefficient use of resources; even after being copied by the destination address station the frame is repeated from node to node all the way to the originator. On the average half of the transmission time is used for frame repetition from node to node after the DA

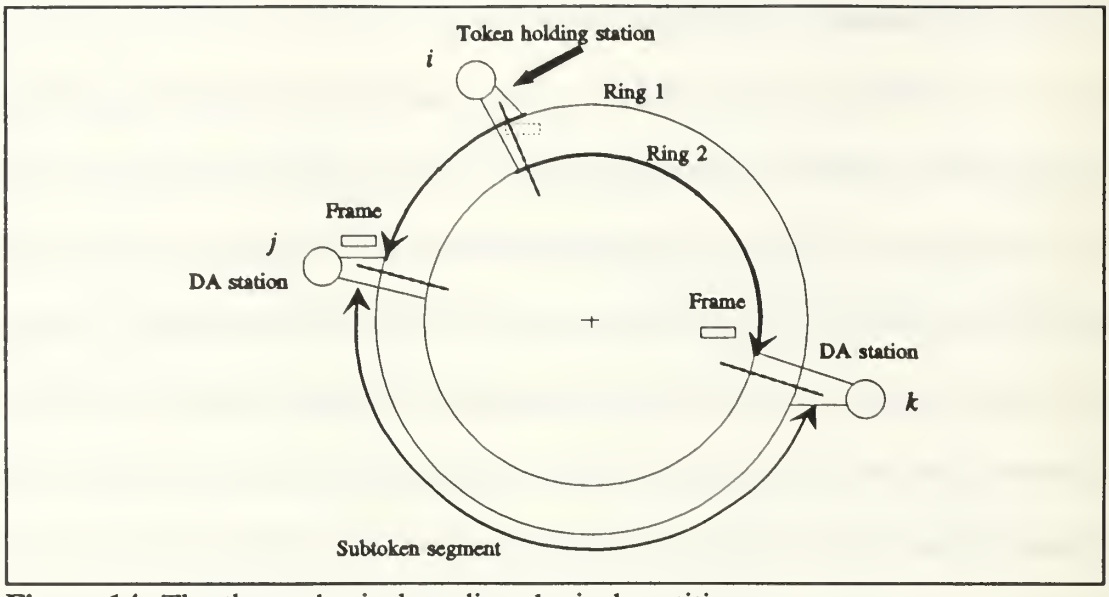


Figure 14: The three physical medium logical partitions

node has copied the frame. The allocation of a second ring which remains unused during normal operation further aggravates this situation of inefficient use of resources. In the improved protocol, with the exception of MAC frames, which need the full ring to perform the Ring Initialization, Claim, and Beacon processes, other frames are not repeated on the ring all the way up to its originator. Instead, these frames are removed by the destination address stations. This procedure allows the use of a freed segment of the ring to be used by any station within that segment concurrently with the main token holding station. The acknowledgment is sent by the destination address station on the opposite ring in a segment that is not being used. As opposed with the acknowledgment embedded in the frame, which is repeated downstream up to the frame originator, the new MAC employs a short fixed-length PDU that goes back on the opposite ring immediately after the frame is copied. This PDU carries no information field but only

the necessary sequences of symbols for the frame originator MAC to provide status report service to other entities.

The concurrent access to the physical medium is controlled and synchronized to avoid collisions. Stations may have many PDUs queued each with a different destination address. Also, frames are variable in length (duration); some can be frames of short duration other can be in the order of a maximum length allowed in the protocol. Therefore, a mechanism to manage the use of space and time is needed. The solution adopted by this protocol is to provide another type of PDU that controls the access of the second transmitting station while the first station is accessing the ring during normal operation. This PDU called "subtoken" is controlled by the main token holding station. This station generates subtoken information for concurrent transmission on the unused segment of the ring. The next subsection presents the subtoken and the acknowledgment format.

3. Protocol Data Units (PDU) Formats

In addition to frame and token formats used in the original FDDI, this protocol adds two other fixed-length PDU formats: subtoken and acknowledgment. The purpose of the subtoken is to grant stations the right to transmit on the unused segment of ring. Therefore, the subtoken contains fields establishing physical limits for transmission on the ring. Its format also includes a field that gives the maximum duration allowed for one or more frames to be transmitted concurrently with the frame of the token holding station.

The acknowledgment is a short fixed-length PDU that returns on the other ring in opposite direction from the data frame. The acknowledgment is sent after the receiver on the ring of the incoming frame copies this frame into its input buffer.

Figure 15 depicts the subtoken format. The starting sequence (PA field and SA field) is the same as in frame or token. The subsequent fields are as follows:

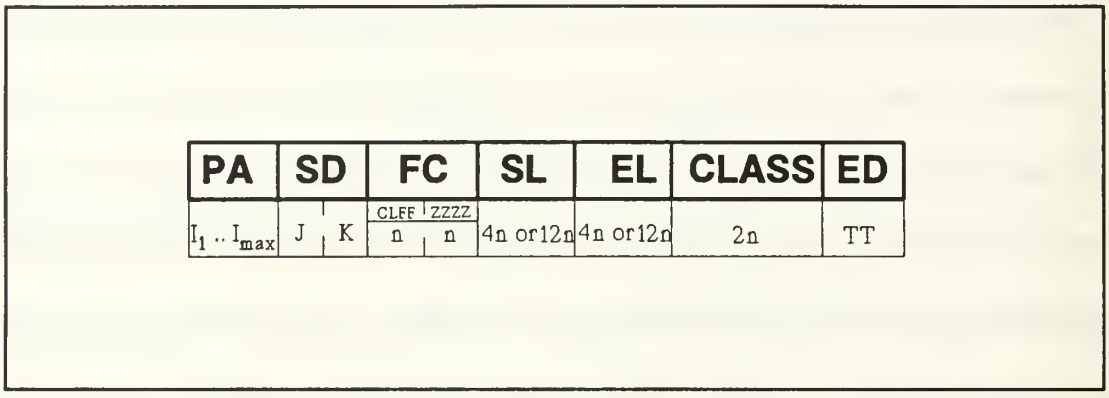


Figure 15: Subtoken Format

- Frame Control (FC) - consists of eight bits (two data quartets) specified by the following bit format: CLFF ZZZZ, where for the subtoken they are 1111 0000.
- Start Limit (SL) - consists of four or 12 data quartet symbols to indicate the address of the station where the subtoken starts to be valid for use.
- Ending Limit (EL) - consists of four or 12 data quartet symbols to indicate the address of the station where the subtoken stops to be valid for use.
- Frame Class (CLASS) - consists of two data quartet symbols to specify an upper bound for the length of the frame to be sent by the station that is holding the subtoken
- Ending Delimiter (ED) - same as in token; consists of a pair of symbols (T) to indicate a subtoken ending. This field is necessary to provide a criteria for acceptance of a valid subtoken. The ED must be met before a subtoken is accepted.

Beginning with the Start Limit address, the subtoken is "captured" or passed on from station to station downstream on the ring until it reaches the Ending Limit (EL) address. If the Ending limit station does not use the subtoken then this station strips it from the ring.

This protocol employs the CLASS field in the subtoken to provide information for ring scheduling on the partitioned segment of the ring. Classes represent time duration of frames. Figure 16 depicts the formation of classes. As the figure illustrates, classes can be a step-wise or sampled discrete function of frame length.

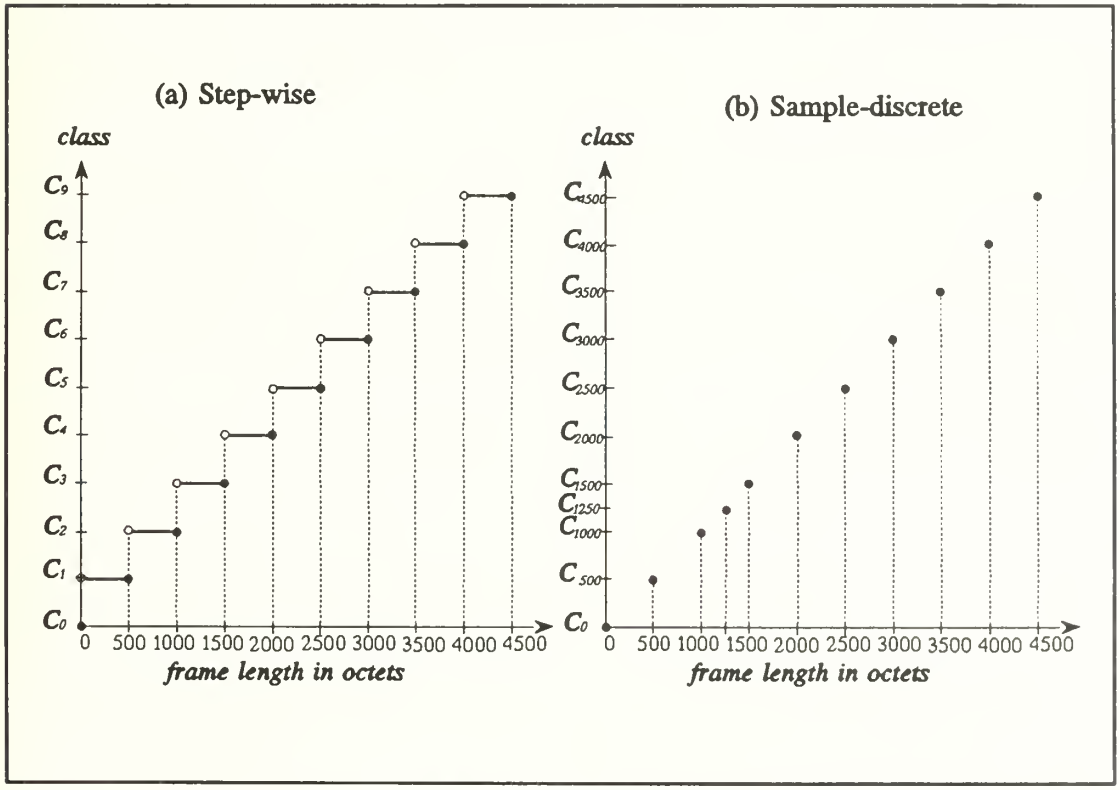


Figure 16: Classes as a Step-wise or Sampled-discrete Function of Frame Length

In the step-wise representation of Figure 16, 10 classes beginning from zero through nine are mapped to intervals of frame lengths in multiples of 500 octets from zero through the maximum frame length of 4500 octets. The third Class C_2 in this example corresponds to a number of symbol times calculated for the interval (500, 1000] of octets. If a step-wise formation of classes is adopted the number of classes could be implementor defined. In this case, the granularity of classes could depend upon the size of files used in the application for the FDDI network. In the sampled-discrete case, there are virtually as many classes as discrete sizes of frames; however, for short frames it is reasonable to consider a mapping to a zero duration class because these frames of short duration cannot be used in the process of concurrent access.

Classes are measured in number of symbol times at 100 Megabits per second. The number of symbol times for each class must be determined for the network to ensure adequate transmission of data concurrently on the ring.

The main token holding station establishes the class of the subtoken according to the length of the two frames which are transmitted simultaneously. The token holding station assigns a subtoken Frame Class that corresponds to an interval of frame length calculated by that MAC station based on the both frames queued for transmission. In the next subsection, algorithms to generate classes for the life of subtoken are discussed. The station that uses the subtoken can only transmit a frame during the extent of that received frame Class.

Any station which address falls within the subtoken limits may gain the right to transmit by capturing the subtoken. However, this may only occur under the following conditions:

- The candidate station may only transmit frames addressed to stations that are physically within the limits established by the subtoken fields SL and EL of that particular subtoken.
- The candidate station may only transmit frames that have length bounded above by the class of frame defined by the subtoken field CLASS of that particular subtoken.

These conditions avoid collisions and allow controlled concurrent use of the partitioned rings.

Figure 17 depicts the acknowledgment format. The acknowledgment follows the same format pattern of a frame in the original FDDI protocol, except that the INFO and FCS fields do not appear. The Destination Address (DA) in the acknowledgment format corresponds to the address of the received frame originator, and the Source Address (SA) corresponds to the station that is forwarding the acknowledgment.

The Ending Delimiter (ED) and Frame Status (FS) fields are the same as in the frame; however, the FS field in the fixed-length acknowledgment consists of exactly the three control indicator symbols for Error Detected (FS.E), Address Recognized (FS.A), and Frame Copied (FS.C). There is no trailing terminate "T" symbol after the FS field symbols since one "T" symbol from the ED field plus three control indicators of FS field form a sequence of symbol pairs required by MAC FDDI standard.

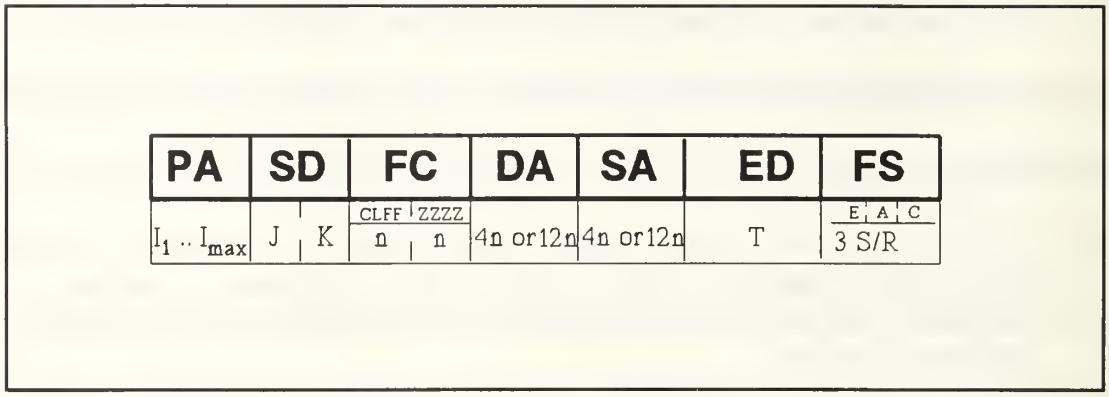


Figure 17: Acknowledgment Format

A different procedure adopted on the process of acknowledgment contributes to still more improvement. This difference is in the way the acknowledgment is routed to the originator. In the original protocol, the receiving station matches the DA field to its own address, copies the frame and sends the acknowledgment inside the frame on the primary ring. In the improved protocol, after the receiving station scans the received Destination Address (DA_r) and a match occurs with its own address, it strips the frame from the primary ring as it is copied to the local entity. After the frame is received and copied the station transmits an acknowledgment back to the originator on the opposite ring. This procedure avoids unnecessary frame propagation and it frees a segment of both rings for the candidate stations to use the subtoken on that part of the rings.

B. THE TIMED-TOKEN CONTROLLED CONCURRENT ACCESS

The format of the additional protocol data units have been described. Provided that information, this section presents a more detailed description of how concurrent access with simultaneous transmissions on both rings can be achieved in this protocol.

The main token holding station is responsible for providing the necessary information to allow other station's MACs to work cooperatively on the dual ring. This is accomplished as follows. Assume that there is synchronous traffic queued for transmission. The traffic is distributed for transmission among the two physical medium using a balance loading algorithm to improve efficiency. There are two queues for this service; one for each ring. The Service Data Units (SDU) are placed in the frames and enqueued for transmission. The process begins when a passing token is captured by a station as in the original FDDI as shown in Figure 18.

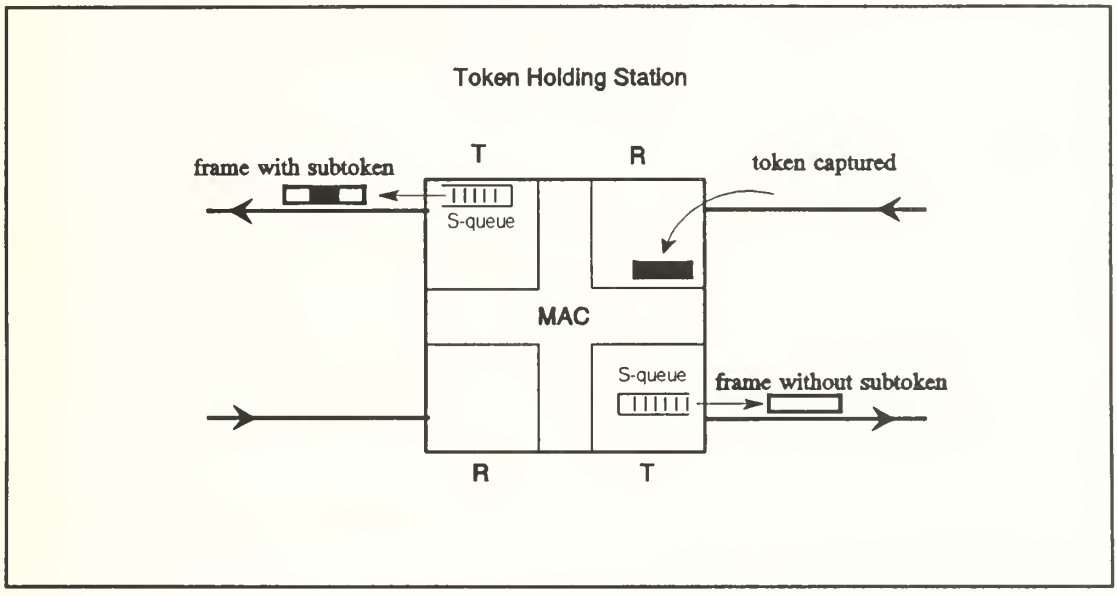


Figure 18: The Token Holding Station Simultaneous Transmission on Both Rings

This station begins transmission of the two head queued frames simultaneously on both rings. One of these two frames will carry additional information to allow a posterior formation of the subtoken as a separate PDU. This frame will be the *frame with subtoken*. The other frame is transmitted simultaneously on the opposite direction *without*

the subtoken information. The MAC Transmitter in charge of creation of the frame with subtoken inserts the Destination Address (DA) from the frame without subtoken, queued on the other ring, in the first eight or 12 symbols of the frame INFO field. The DA will become the subtoken Ending Limit (EL) in the subtoken PDU format. Furthermore, the Transmitter will insert immediately after these symbols, the duration measured in symbol times at 100 Megabits per second that a station will be allowed to transmit when using the subtoken. This duration corresponds to a CLASS value in the subtoken format. Figure 19 depicts the contents of the frame with and without subtoken information. The figure shows the formation of the subtoken contents. The two types of frames are summarized as follows:

- Frame with subtoken - issued by the token holding station only. It carries the subtoken information and the Service Data Unit. When the protocol is in the "WRAP" mode the token holding station does not issue frames with subtoken.
- Frame without subtoken - issued by either the token holding station or the subtoken holding station. This frame is the same as the employed by the original FDDI.

Once the frame with subtoken reaches the destination address (DA) station the subtoken is ready to be used by any station within the ring segment limits beginning from this DA station up to a Ending Limit (EL) station established by this field in the subtoken format. Therefore, the first candidate station to use this subtoken is the frame destination address station itself.

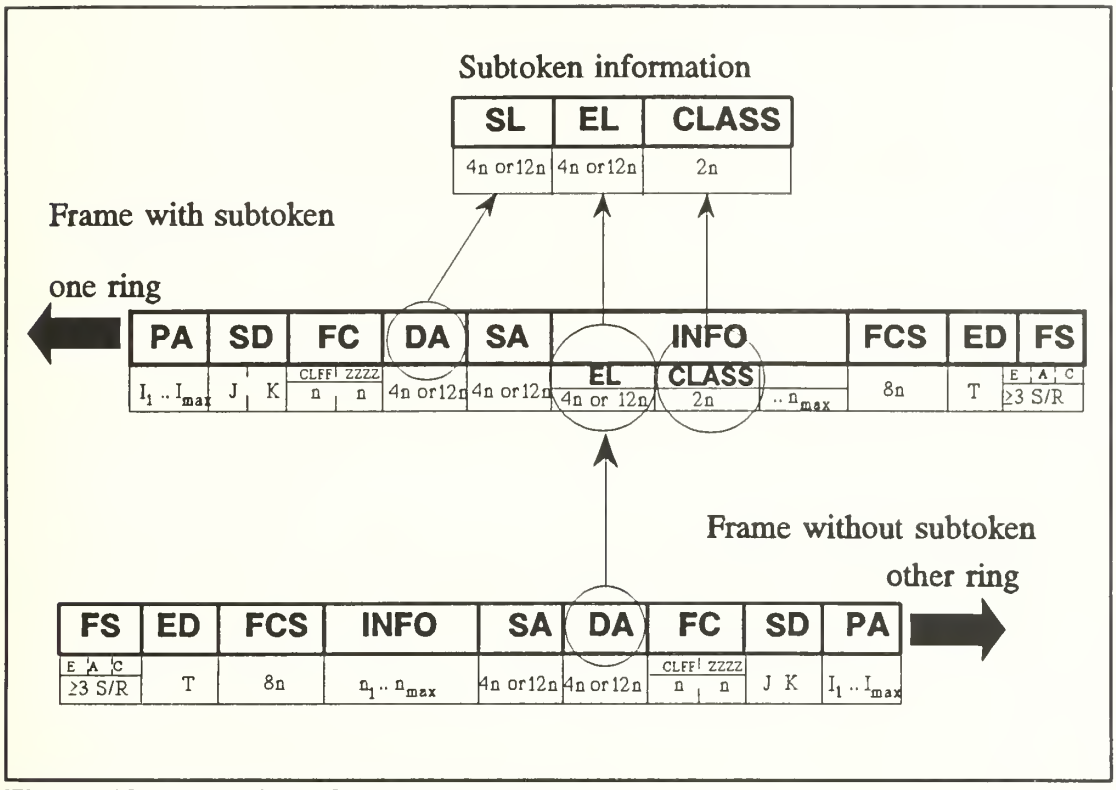


Figure 19: Formation of Subtoken Contents

1. Algorithms to Generate Subtoken Duration (Class)

Two algorithms to generate the duration of the subtoken are presented. The first algorithm is simpler, generates less number of subtokens; consequently, imposes less overhead. However, if head queued frames on each ring differ substantially in length one segment of the ring may become unusable waiting until completion of other ring frame transmission. The subtoken duration in this algorithm is calculated based on the largest of the two frames. The second algorithm has one step more than the first one, generates more subtokens and can make more use of the ring; however, adds more overhead. The next paragraphs will present the initial assumptions and the two algorithms with practical illustrations of their first iterations.

Assume that Class is the same as frame length. Also, assume that a station has captured a token and is ready to initiate its simultaneous transmissions on both rings with the load balanced on both queues. The first algorithm is described as follows:

- Get the Class from the head of the queue for the primary ring and the Class from the head of the queue for the secondary ring.
- Compare these last two Classes. If they are equal take this value; otherwise, choose the largest of the two Classes. The value chosen becomes this subtoken duration (Class).
- After completion of frame transmission go to the first step. Repeat the process until either the station completes its transmissions or the synchronous bandwidth allocation for this station transmissions terminates or the THT expires.

Figure 20 illustrates the first two iterations of the algorithm. The figure shows a bar chart that represents frames enqueued for transmissions on both rings and the resultant duration of the subtoken. In addition, the figure shows a dual ring architecture with 20 stations and the simultaneous transmission carried out by station number one, which holds the token. Numerical examples are shown to enhance the illustration.

In this algorithm the subtoken can go in either direction; however, it is assumed by default that it goes in the same direction of the token. In this example, the first subtoken can be used by any station beginning from address three up to address 15 and it allows transmissions within these limits that take a maximum of 1500 octets. Note that this duration corresponds to length of the frame queued on the primary ring. Since the frames have unequal length, one segment will complete its transmission earlier than

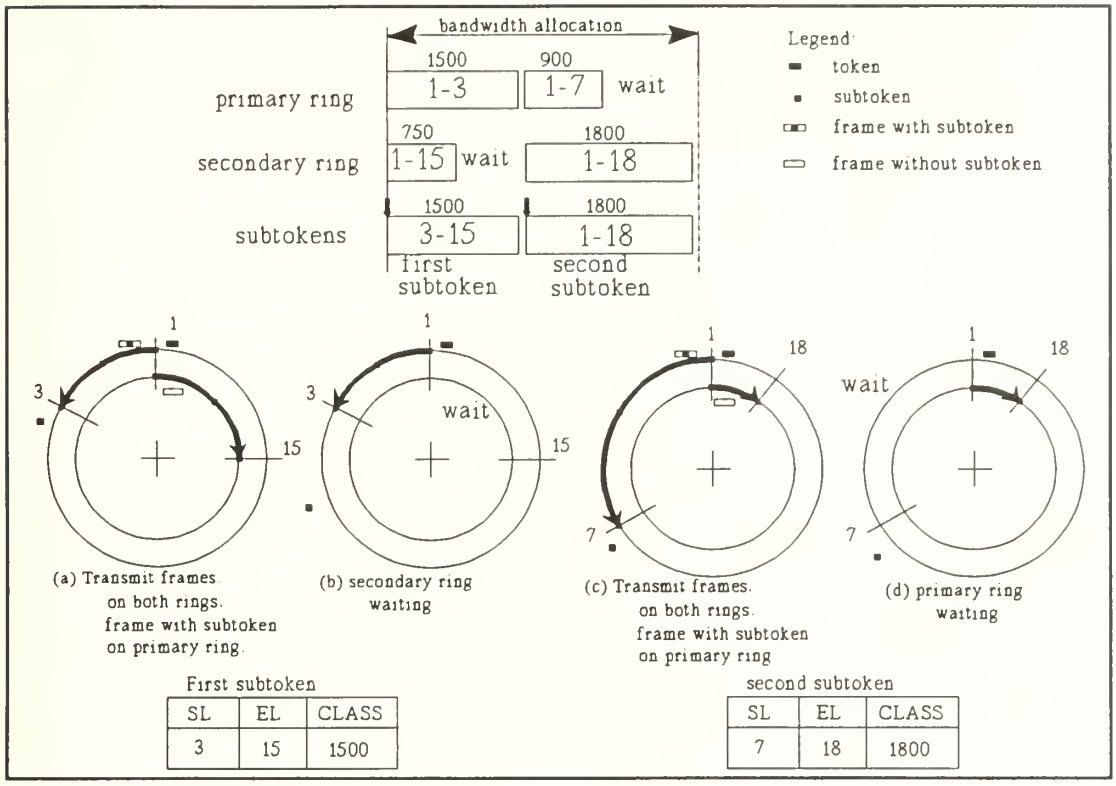


Figure 20: Illustration of the first Algorithm

the other; consequently, this segment will wait for the completion of the other when both reinitiate a new simultaneous frame transmission. The figure shows the four timing steps on the dual ring operation during the allocated bandwidth.

The second algorithm makes full time use of both segments. It is described as follows:

- Get the Class from the head of the queue for the primary ring and the Class from the head of the queue for the secondary ring.
- Compare these last two Classes. If they are equal take this value; otherwise, choose the shortest of the two Classes. The value chosen becomes this subtoken duration (Class).
- Save what's left over from the last comparison as a last Class.

- Get the next Class from the queue for the ring which had the last shortest Class and go to the second step. Repeat this process until either the station complete its transmissions or the synchronous bandwidth allocation for this station transmissions terminates or the THT expires.

Figure 21 illustrates the application of this algorithm for the same data presented in Figure 20. In this algorithm, the subtoken goes on the ring that last terminate a frame transmission. An exception for this condition is the first simultaneous frame transmission or when there is a match such that both rings reinitiate their frames transmissions at the same time. In this case, the subtoken can go in either one or another ring.

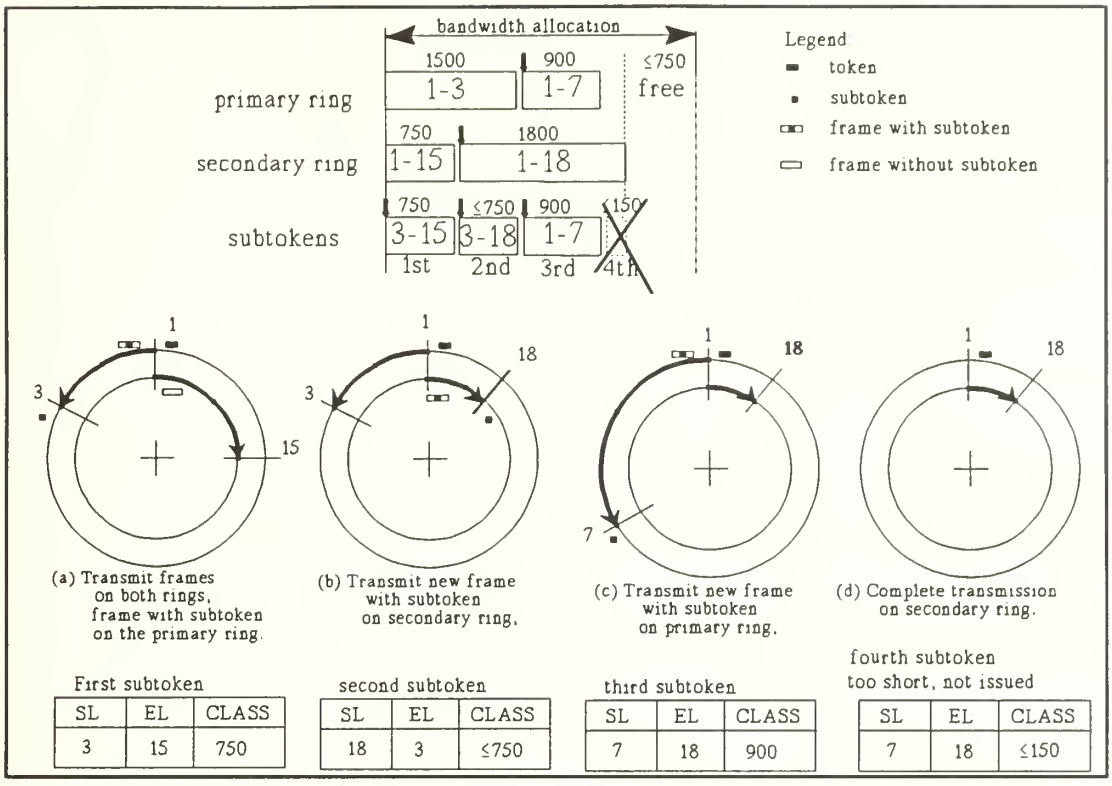


Figure 21: Illustration of the Second Algorithm

Figure 21 shows that the second algorithm increases utilization of both rings since one ring does not wait the opposite ring to complete the transmission of its current frame. As the illustration shows, there is a saving in the bandwidth for simultaneous transmission if compared with the previous example. Also, the illustration shows that this algorithm increased the number of subtokens generated by the token holding station. However, this algorithm leads to less use of the subtoken ring segment. When the calculated duration of the subtoken becomes too short the token holding station might not issue this subtoken; consequently, some available time on this third segment of the rings is not used. In addition, this algorithm imposes more overhead as a result of these decisions and the increased number of subtokens with short durations.

This thesis uses the first algorithm in the formal specification of the improved protocol. This decision was primarily based on the simplicity of the first method. Both algorithms increase throughput and a more complete analysis such as the use of a simulation model would enhance the list of pros and cons for each method.

2. Destination Address Station Actions

The MAC station corresponding to the frame destination address (DA) is responsible to take several actions in this process of concurrent ring access. First, it scans the field contents of the incoming frame and checks the Frame Control (FC) to verify the type of PDU; note that in this protocol the FC field is also used to differentiate frames that carry subtoken information from frames that do not include it. Second, the station matches the frame DA with its own address and begins to copy the frame into its receive buffer while taking actions to remove this frame from the ring. If the frame contains the subtoken contents then the station may use the subtoken or not. In the case of subtoken use, the station immediately begins transmission of its PDU downstream on the ring while receiving the data upstream from the token holding station. The timing mechanism of concurrent access shall work such that it is expected that the traffic downstream terminates before the traffic upstream and both stations had received an acknowledgment back on the opposite ring. Finally, in the case of unusable subtoken the station immediately extracts the subtoken contents from the frame and issues the subtoken as a separate PDU downstream on the ring to be used by another station on the freed ring segment.

Stations pass the subtoken if they can not use it. The subtoken passing process continues until it reaches the last station on the unused segment when this station finally removes the subtoken from the ring. If the subtoken is used before it reaches the ending limit the station that uses the subtoken is responsible for its removal from the ring (i.e., the subtoken is not reissued). The subtoken can be used only once.

3. Reconfiguration

An important issue to discuss is the reliability provided if the changes are implemented in the FDDI protocol. Basically, FDDI provides fault tolerance with a dual counter-rotating ring because of its "WRAP mode," in which stations reconfigure to isolate a serious ring or node failure. In this case, the dual logical ring becomes a single path allowing communication to continue.

The improved protocol also provides this feature but with the loss of enhanced ring utilization. "WRAP" is a property of a station in the original FDDI; however, an overall network-controlling function can monitor the network to detect station's wrap. The stations on either side which reconfigured to isolate the failure notify SMT and a global variable **Wrap** is set to true. The stations recognize that only one logical ring is active and the stripping of frames revert to the original FDDI with the acknowledgment sent inside the frame.

IV. FORMAL SPECIFICATION

This chapter presents the formal specification for the Media Access Control (MAC) protocol developed in this thesis using *Systems of Communicating Machines*.

A. BENEFITS OF A FORMAL SPECIFICATION

One of the main purposes of a formal specification is to provide enhancements in its interpretation. One of the problems pointed out in the interoperability issue is that the interpretation of the standard itself is ambiguous [Ref. 14]. Among other problems, the following example was given:

> It appears to be possible to take in the symbol sequence I J I, which is an error condition, and still conform to the FDDI standard. This causes oscillations of the ring.

A formal model such as the one used in this thesis helps to identify and isolate problems of this nature. To illustrate this specific case, the model specifies a MAC receiver checking for a strip on the incoming symbol sequence which forms the SD field of the PDU. Since there is an "I" symbol after the "J" symbol it means that a transition *Strip on SD* holds true on the Receiver. (see MAC Receiver state diagram and transition table). The MAC Receiver sends a signal "Idle" to the Transmitter and enters the AWAIT SD state. The receipt of "Idle" signal from the Receiver enables a transition *Tx Idle Symbols* on the Transmitter side which enters the IDLE state. Symbols are checked one by one leaving no ambiguities.

The model *Systems of Communicating Machines* can also be applied to the specification of the protocol physical layer and in this case the sequence of operations would consider the code bit level. This provides a better document interpretation and contributes to interoperability in multivendor computing environments.

The FDDI Media Access Control standard presents subclauses containing the MAC structure functional specification. The MAC structure defines two asynchronous processes that work as co-operating state machines, called the MAC Receiver and the MAC transmitter. The standard presents the overall processes operation in the form of state diagrams and attached notes. These diagrams show the transitions that take place between the states. In addition to state diagrams the standard uses prose in the specifications, although it states that the state diagrams shall take precedence in the event of any discrepancy. The standard provides all the specifications but lacks the use of a formally defined model in its protocol specifications.

The MAC specification used in this thesis has some similarities and differences from the MAC specification used in the FDDI MAC standard. They are similar in that both use processes of co-operating machines represented in state diagrams. They use the similar terminology as defined in the previous subclauses of the MAC and other FDDI standards. They differ in that the specification presented in this thesis uses a formal model. The model is used to specify the entire protocol; however, it also applies to hierarchies of small independent modules such that it captures the aspects of the protocol behavior and structure relevant to the development team. Furthermore, the application of the formal model forces a much more detailed consideration of the protocol which

makes the functional specification more precise. This technique provides a refinement of the MAC protocol specification.

The formal specification of the Media Access Control protocol contributes to enhance the standard functional specification in several ways:

- Provides precise control over each protocol module or state behavior.
- Improves understanding of the protocol functionality. Modules are understandable in isolation. For a complex protocol such as FDDI, this is a desirable property.
- Reduces documented protocol ambiguities. The decomposition into small independent modules provides a clear relation between each module and the protocol functional specification.
- Confines changes in design decision to a single module. If the protocol needs changes then this will not depend on the entire specification.
- Provides means to conduct a protocol analysis for its correctness. Proofs for protocol correctness can be established.
- Easier to test the protocol. Decomposition and modular specification simplify testing of each protocol module.

The decomposition of bigger states into a hierarchy of smaller states obtaining a precise control of the a state behavior simplifies the description, protocol verification and implementation. The control over each state is a good feature for identification of malfunctions within each state. This allows development of protocol test procedures, verification and error checking. The understanding of protocol functionality is improved, since at any given moment a transition from one state to another can be checked by following the preceding history of operations in the hierarchy of states and the transition table logical sequences.

The formal definition of each logical operation in separate states avoids documented protocol ambiguities and improves interpretation for implementation purposes. The functional specification for MAC frames in the Receiver are treated differently from the specification of LLC frames because some actions will not be the same for both types of frames. These are treated in different protocol modules. Even states that perform similar functions receive individual treatment within each branch of the transition state diagram.

One relevant question is how deep the decomposition into smaller states will be? Does the number of states and transitions increases to a point such that it becomes unworkably large? The decomposition of states will go up to the level where no ambiguity is left. All the internal events described in the standard documentation which cause an action are considered. A modular decomposition allows an easier implementation of large systems.

Since the specification goes to a deep level of details in terms of transitions and states, it is impractical to explain all these details in one chapter. The reader is encouraged to study the MAC FDDI standard specification. The changes implemented in FDDI to achieve the improvement will alter some modules of the original FDDI; however, the overall functions are the same.

B. MODELING THE IMPROVED PROTOCOL

1. Notational Conventions

The following notation is used in the transition tables:

- $\vee$: logical or

- $\wedge$: logical and
- $\neg$: logical not
- $\ni$: such that
- $\in$: member (element)
- A$\leftarrow$B : places the current value of B into A

In the state diagrams, states are represented by bubbles and transitions by arrows. A state is numbered by a letter-number and is named with capital letters which appear inside the bubble. The letter-number is organized within each module of the diagram such that the letter corresponds to the first letter of the module name and the number corresponds to the number of states within the module. A transition is named with lower case letters (except for the first letter or acronyms used to form the name) and appears in the right side of the arrow between two states.

2. The Communicating Machines

Figure 22 depicts the communicating machines. There are four machines that work cooperatively on both rings. At the node **i** the machines T_i1 and R_i1 are respectively the Transmitter and Receiver on ring 1 while T_i2 and R_i2 stand for Transmitter and Receiver on Ring 2. Machines in one ring are symmetric with machines of same name on the other ring. This allows for the MAC to choose either one or the other ring to circulate the main token. Also, this property allows the representation of states for only one Receiver and only one Transmitter.

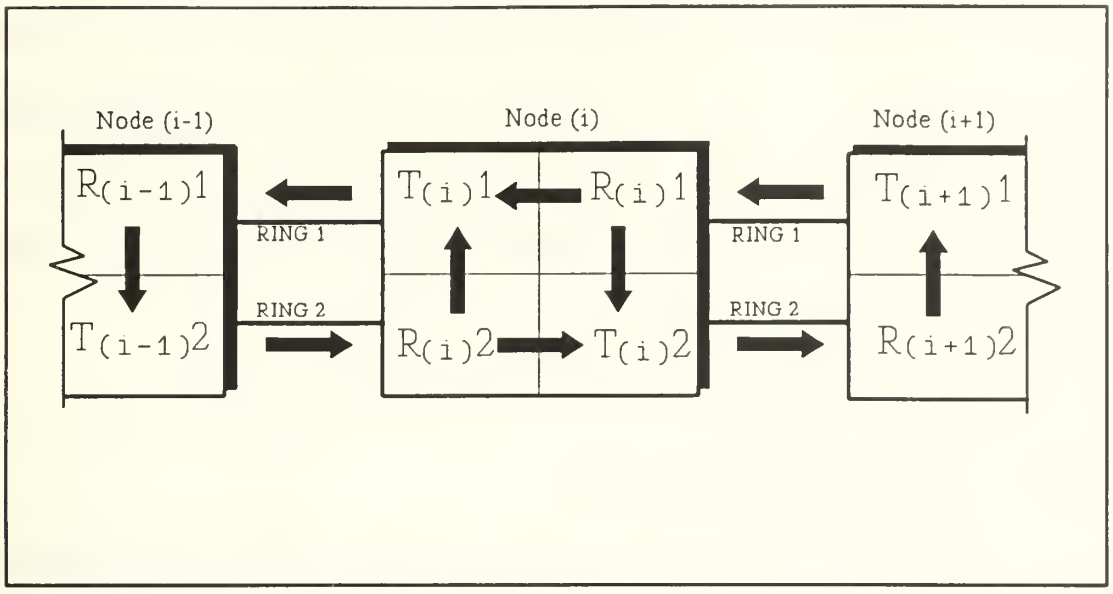


Figure 22: The Four Communicating Machines of a Network Node

Receivers and transmitters behave differently in the communication process. While a Receiver communicates simultaneously with both Transmitters a Transmitter can only communicate with a Receiver on the same ring. A simultaneous communication of a Receiver with both transmitters means that actions taken by a Receiver are visible to both transmitters. However, the Media Access Control establishes the control of the action based on the nature of this action and its originator.

3. Modeling The Interface Operations

There are four operations on the FDDI network specified by this formal model. These are receive, receive-copy, repeat, and transmit symbols.

Figure 23 illustrates the logical operations carried out in a single-MAC-dual-PHY interface. For each ring there is one input variable of type *buffer* denoted as PH_Indication(symbol) and one output variable of type *buffer* denoted as PH_Request(symbol). These are shared variables through which the machines

communicate. Each name is an analogy to the service primitive provided at the interface MAC/PHY. **Symbol** is a pointer associated with these buffers. The pointer locates a symbol on the array of symbols and is sequentially incremented at each new symbol arrival on the receiver side or new departure on the transmitter side. In this model, a variable of type *buffer* is specified by:

type *buffer*: array[1..MFL+1] of symbol;

where MFL is the maximum frame length.

Each transmitter maintains other variables of type buffer dedicated for transmission of its PDU queued. The synchronous service is provided with the S-buf(symbol) and the asynchronous with the A-buf(symbol). Each receiver uses the R-buf(symbol) to copy the PDU that matches with the station's address.

In the *receive symbols* operation, bits from the ring enter the interface through the line receiver into the PH_Indication(symbol) buffer in a serial fashion. These bits form the symbols that are read in the interface PHY/MAC. The symbol pointer is incremented to the next symbol and a symbol counter is maintained. For the receive symbols of a frame the operation is represented as follows:

- PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ };
- symbol ← symbol + 1;
- symbol_ct ← symbol_ct + 1;

In the *receive-copy symbols*, in addition to the previous operation bits are copied by the Receiver into its receive buffer. This operation is represented as follows:

- PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ };

- Rcv-buf(symbol) ← PH_Indication(symbol);

- symbol ← symbol + 1;

- symbol_ct ← symbol_ct + 1;

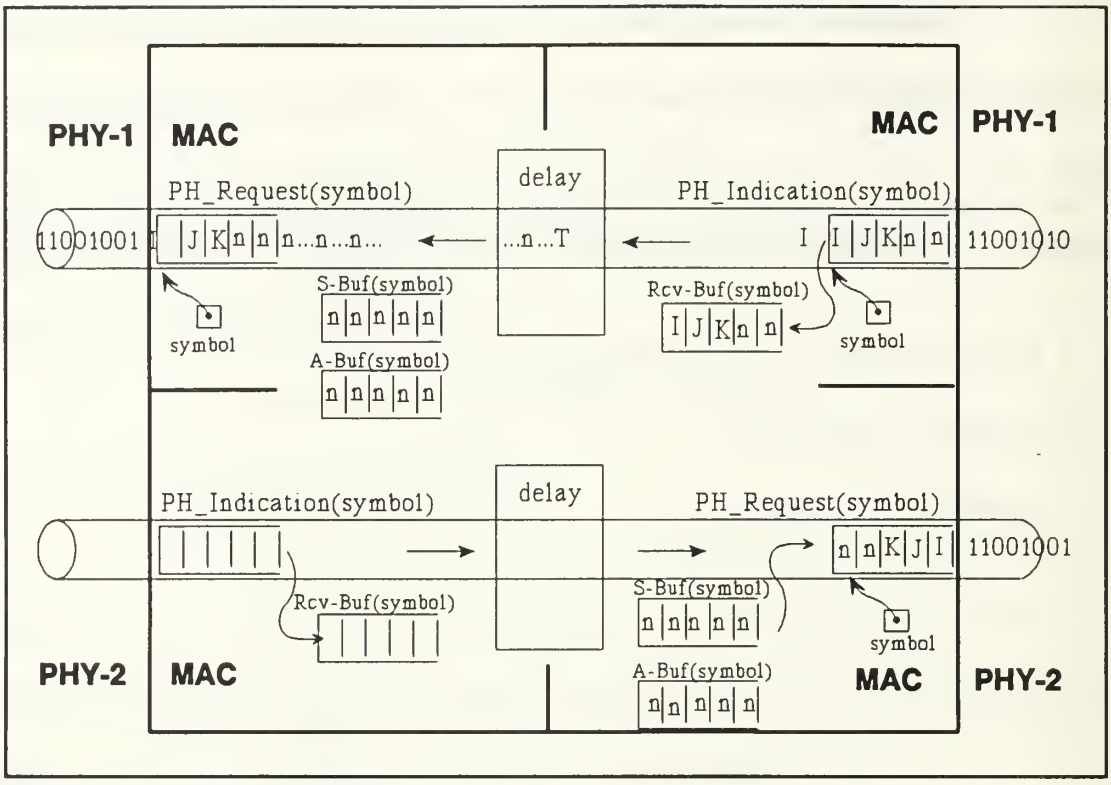


Figure 23: Logical Operations

In the *repeat symbols* operation, after a delay these bits are retransmitted by the Transmitter over the ring from the interface MAC/PHY either unchanged or after some modification. The representation for repeat token symbols is as follows:

- PH_Indication(symbol) = { PA[I_1..I_n], SD[J,K], FC[CLFF,ZZZZ], ED[T,T] };
- PH_Request[symbol] ← PH_Indication(symbol);
- symbol ← symbol + 1;
- symbol_ct ← symbol_ct + 1;

In the *transmit symbols* operation, bits are transmitted serially from the interface over the ring. These bits come from the appropriate buffers (synchronous or

asynchronous) and sent to the line driver for transmission out of the PH_Request buffer. The representation for a transmission of synchronous frame symbols is as follows:

- S-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, FC_x[CLFF,ZZZZ], DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] }
- PH_Request[symbol] ← S-buf[symbol];
- symbol ← symbol + 1;
- symbol_ct ← symbol_ct + 1;

4. The MAC Receiver Operation and Specification

The MAC Receiver machine performs several operations upon receipt of symbols of an incoming PDU. It receives the information from the ring, detects format errors on the field sequences which form each type of PDU, checks frame validity criteria and sends appropriate signals to the Transmitter.

As the symbols of the incoming PDU arrive in a serial transfer from the Physical Layer (PHY) to the Media Access Control Layer (MAC) the MAC receiver machine scans the input, in order to take the appropriate action as required. Figure 24 illustrates how the MAC receiver performs these operations.

The PH_Indication(symbol) appears on the left-hand side of the figure. The right-hand side shows the format of the incoming PDU and the sequence of transitions that occur whenever the corresponding enabling predicate associated with each transition holds true. For example, when the first Idle symbol of the incoming PDU arrives at the PH_Indication(symbol) buffer a *Signal Start* transition occurs and the Receiver takes the

appropriate actions. In general, an action can be described as a signal sent to the Transmitter; a flag setting; a variable status change; or an integer type variable increment such as counter, timer, or pointer increment. In this example, one of the actions corresponding to the *Signal Start* transition is the increment of the symbol pointer to the next symbol of the sequence. In this sense, the Receiver continuously scans the incoming stream of symbols up to the last symbol that forms the complete PDU.

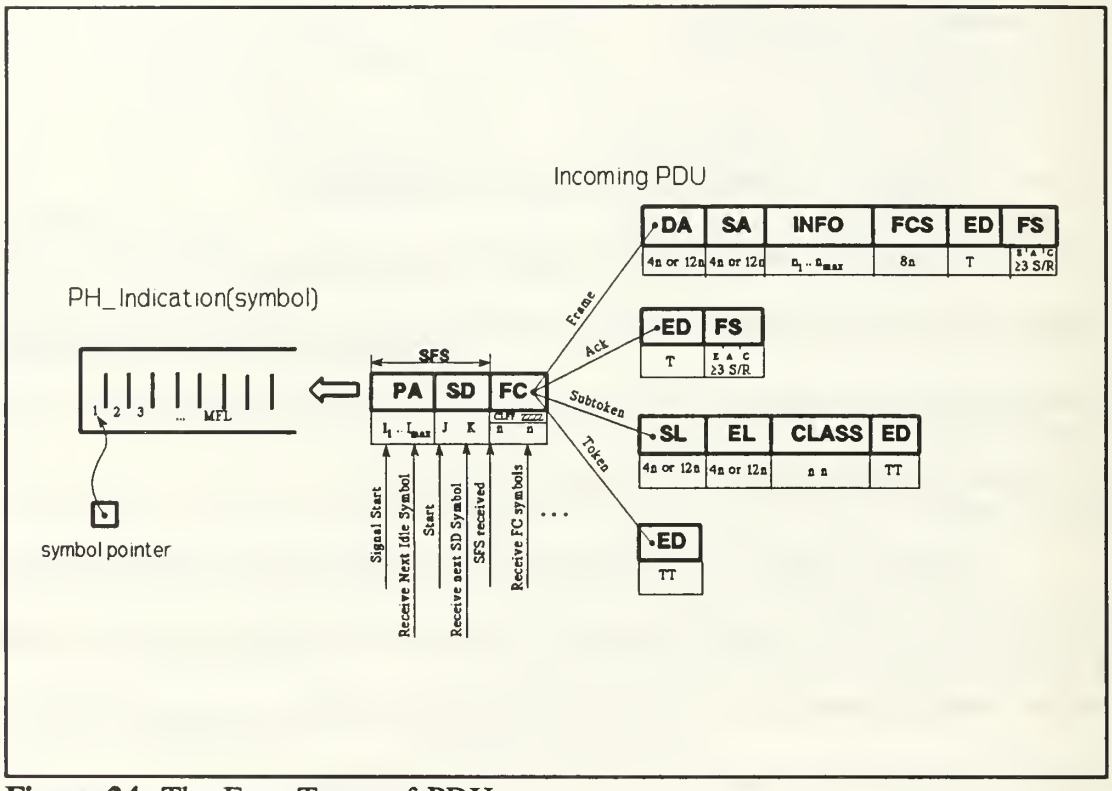


Figure 24: The Four Types of PDU

This specification is more detailed than the standard because in this model every PH_Indication is considered. This corresponds to a decoding of every symbol from the physical layer. For example, when the Receiver departs from the LISTEN (R0) state to the AWAIT SD (R1) state due to the arrival of the first idle symbol, this model will

also consider subsequent idle symbols expressed by a transition called *Receive Next Idle Symbols*. This transition will continue to occur for every other incoming idle until the first symbol "J" of the starting delimiter arrives. Therefore, the model allows a complete and continued checking of symbol sequences, which is exactly what the MAC protocol requires for its correct functionality. This strict control over symbol sequences leaves no ambiguities regarding the protocol acceptance of symbols. The same reasoning is followed for other sequences of symbols until a complete PDU has been received.

As Figure 24 illustrates, the PDU for the improved protocol has four types of formats. Each format adopts the same Starting Frame Sequence (SFS) as FDDI. Immediately after the SFS there is the Frame Control (FC). By scanning the FC field the Receiver takes one out of four types of PDU to follow through on its searching. Similarly, the Receiver state diagram shows four main branches, namely; **Frame**, **Ack**, **Subtoken**, and **Token** (as in Figure 24).

a. The MAC Receiver State Diagram and Transition Table

The MAC Receiver State Diagram forms a closed cycle. A hierarchical decomposition and a history on the sequence of operations is observed in each part as the receiver performs the checking on the incoming PDU. These operations cover all conditions that must be met in the receiving process of the protocol.

Figure 25 depicts the complete MAC Receiver State Diagram. The purpose of this figure is to provide a picture of the MAC receiver diagram. This complete diagram is formed by a set of smaller diagrams contained inside the rectangles which are presented in separate figures. There are 10 smaller diagrams in the Receiver

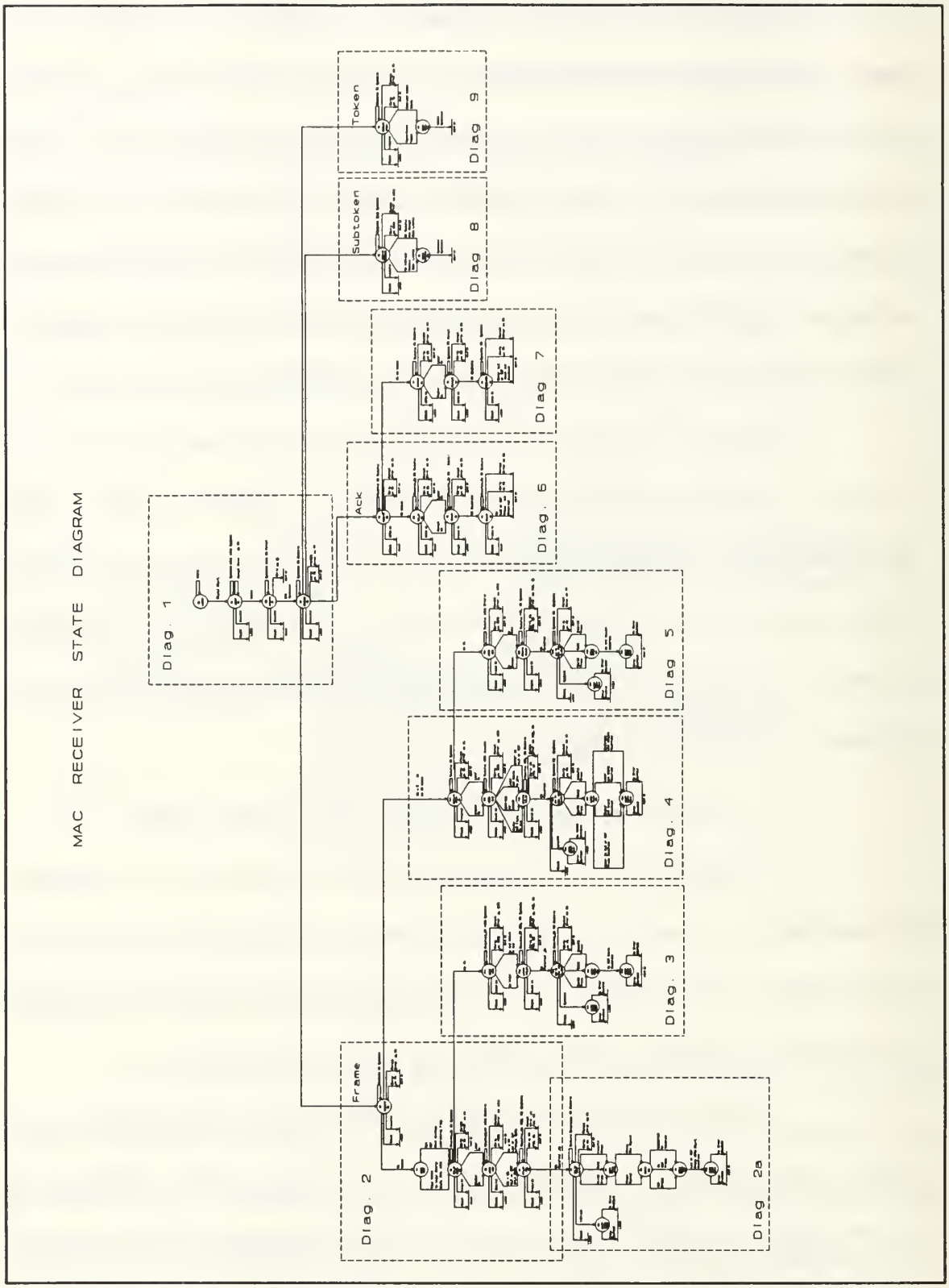


Figure 25: MAC Receiver State Diagram

state machine.

The MAC Receiver state diagram presents the receiving process in five main modules. The first module is represented by Diagram 1 (Figure 26) which corresponds to the Receiver scanning Starting Frame Sequence (SFS) and the FC field of a PDU. After the completion of the FC scanning the Receiver can take four possible transitions in which it looks for frame, ack, subtoken, and token. These are the other four modules of the MAC Receiver State Diagram. The Ack module has two diagrams and the Subtoken and the Token have 1 diagram each.

The Frame module includes Diagrams 2, 2a, 3, 4, and 5. Note that unlike the specification in the original FDDI standard, which does not cover this level of diagram specification, in this decomposition into smaller states the Receiver checks for arrival of different entity frames. A frame which arrives at a station with the Destination Address (DA) equals to its own Source Address (SA) is a MAC Claim Frame. In this case, the station has received its own Claim Frame. The path of transitions goes from Diagram 2 to Diagram 3. The station will copy its own frame into its receive buffer and sends a signal **My_Claim** to the transmitter which strips the frame and begins the process of issuing a token. On the other hand, if a frame arrives with DA not equal to this station address or DA null then the transition path goes from Diagram 2 to Diagram 4 (*DA = 0 or No DA Match*). In this case, a further check is done in the SA field of the frame to verify if this frame is the station's own Beacon Frame; if it is then the transition goes to Diagram 5, the transmitter will be signalled with **My Beacon** and that MAC will attempts to recover the ring. Otherwise, the path goes straight

through Diagram 4 and the frame can be either an upstream Source Address Beacon or a downstream Destination Address LLC, SMT, or MAC frame. This frame will be repeated downstream on the ring by the transmitter.

Furthermore, if a frame is a LCC or SMT frame which was sent by another station to this station then the transition path goes from Diagram 2 to Diagram 2a. This frame will be copied and for this protocol the Receiver will send an **Ack_Frame** signal to the Transmitter on the opposite ring, which in turn will acknowledge the frame. In this case, the Receiver sets the appropriate flag to indicate successful copy of the frame received. The purpose of this brief description of the **frame** module is to show how this specification **avoids ambiguities** and provides a **precise control** over the receiving process of a PDU. Different types of frames during different phases of the receiving process imply different actions which are considered in separate modules. An example of a complete path of transitions in a frame receiving process is given below.

(1) The Destination Address Station Receives a LLC Frame with Subtoken

A complete cycle path of transitions for receiving one LLC *frame with subtoken* information will be given. The path begins in Diagram 1 goes through Diagram 2 and 2a, and finally reaches Diagram 1 for another frame. The frame is assumed to arrive without any format error; is a complete frame (i.e., not a remnant), and no *MAC Reset* transition or PH_invalid are expected. The Receiver copies the frame, checks the validity, and sends the appropriate signal to the Transmitter on the opposite ring to carry out the acknowledgment process.

- R0 *Signal Start* R1 *Receive Next Idle Symbols* R1 *Start* R2 *Receive Next SD Symbol* R2 *SFS Received* R3 *Receive FC Symbols* R3 *Frame* F0 *Receive DA Symbols* F0 *DA Match* F1 *Copy Frame to Local Entity (LLC)* F2 *Receive_Cp SA Symbols* F2 *Lower SA* F3 *Receive_Cp INFO Symbols* F3 *DA=MA LLC With Sbtk* F4 *Receive_Cp FCS, ED Symbols* F4 *ED Received_Cp* F5 *Receive Determine FS Symbols* F5 *Valid Frame* F7 *Frame Copied* F8 *Ack Frame* F9 *LLC Frame with DA=MA Received* F10 *No Error Counted* R1

A finite number of transition paths can be established and used in the chain-reaction arguments process to prove the protocol correctness. In Chapter V a proof for the previous example is provided.

The other modules, from Diagram 6 to Diagram 9, follow the same reasoning in the process of receiving a PDU. The acknowledgment, subtoken, and token modules are simpler than the frame module since these PDUs are of short-fixed length and carry a unique type of information. Similar paths of transitions to receive these PDUs can be established.

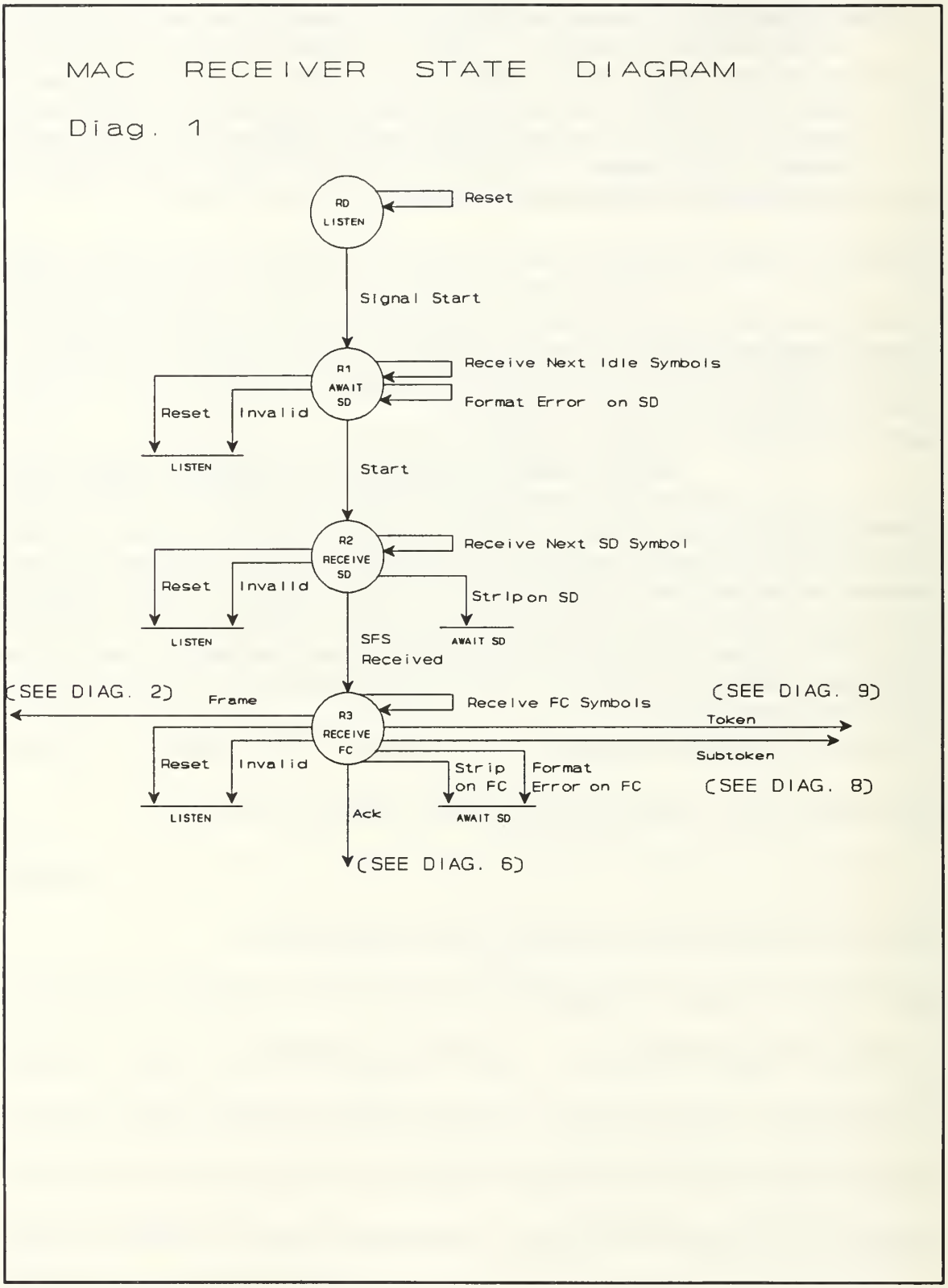


Figure 26: Receiver Diagram 1

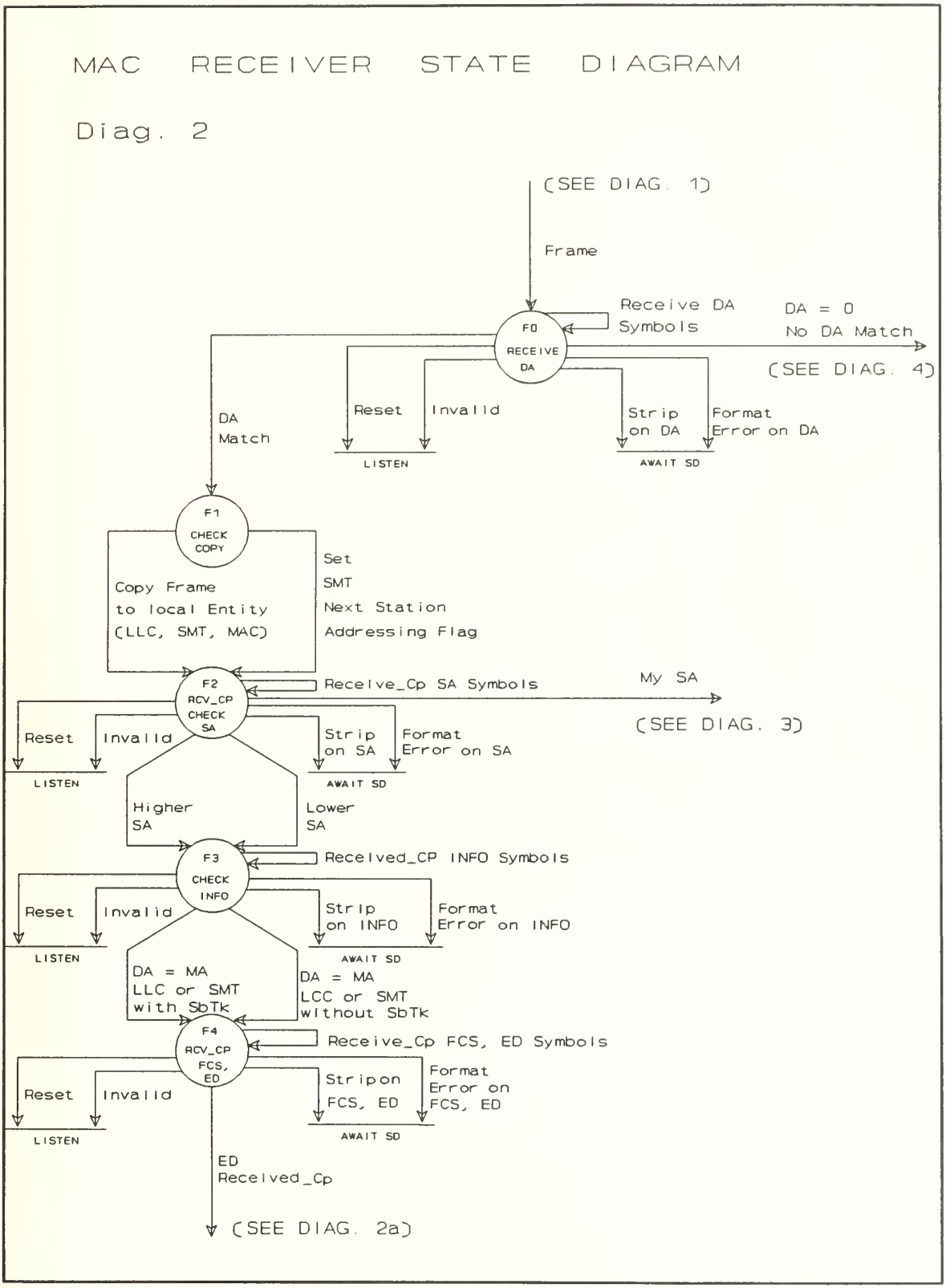


Figure 27: Receiver Diagram 2

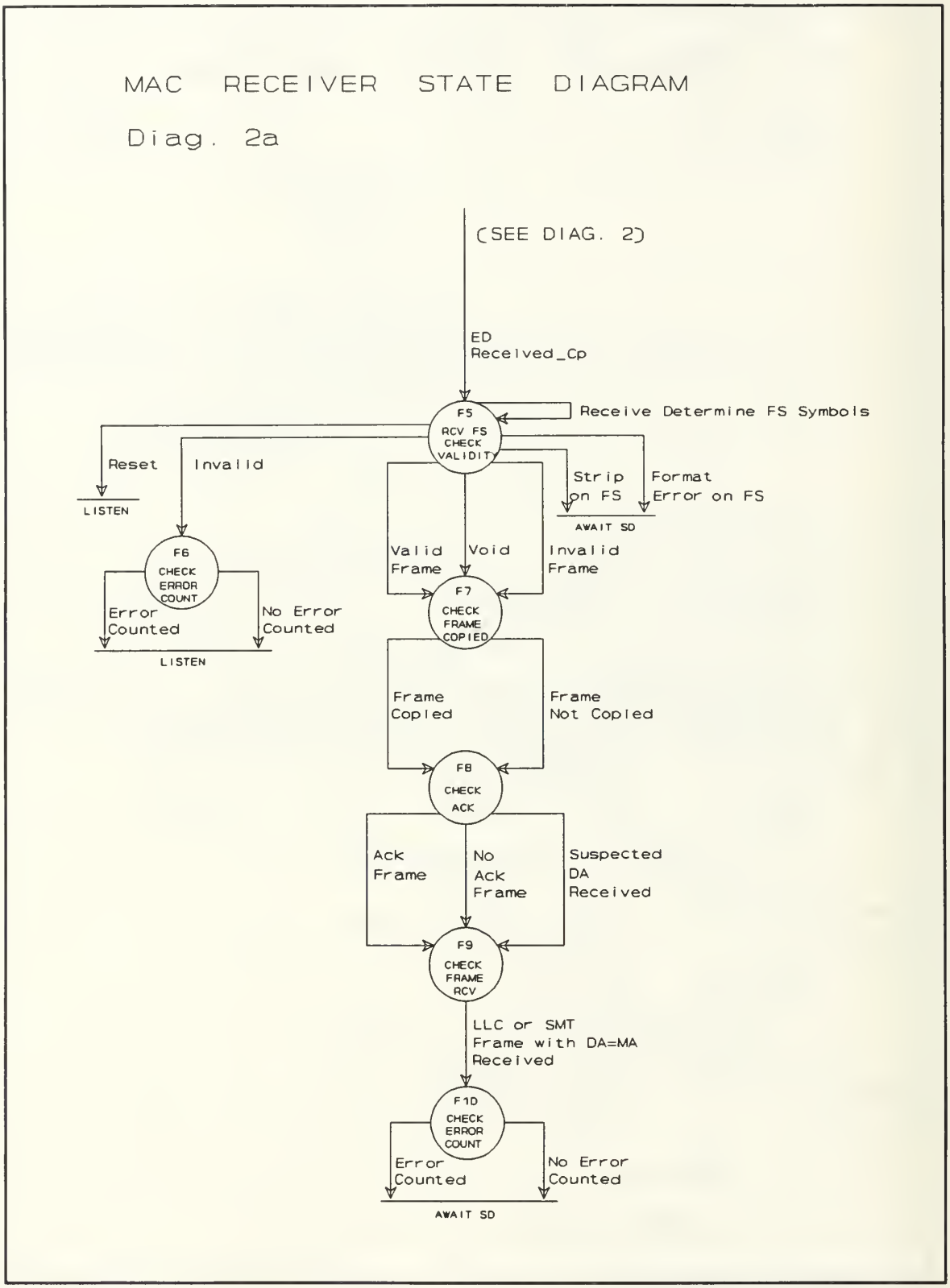


Figure 28: Receiver Diagram 2a

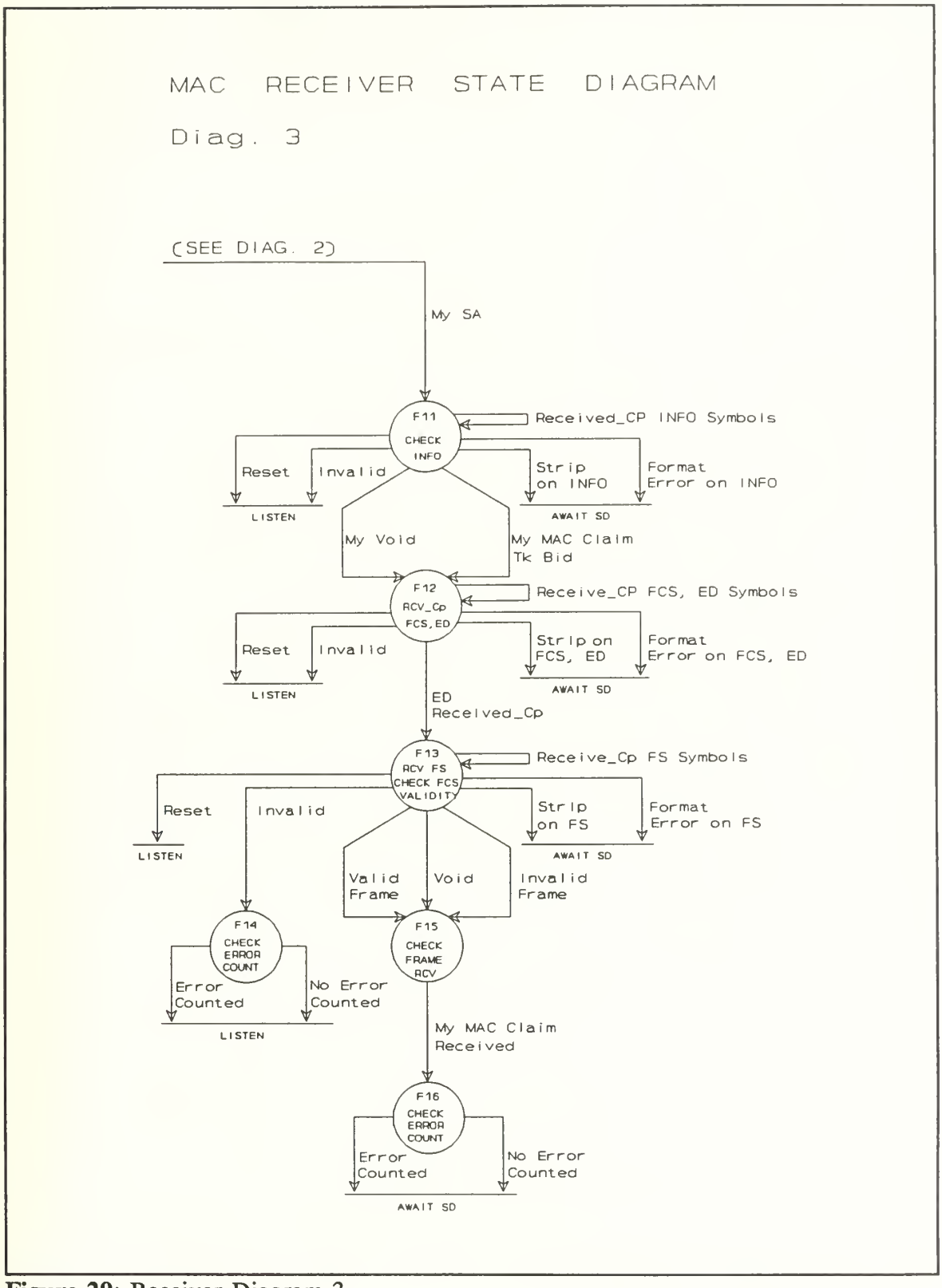


Figure 29: Receiver Diagram 3

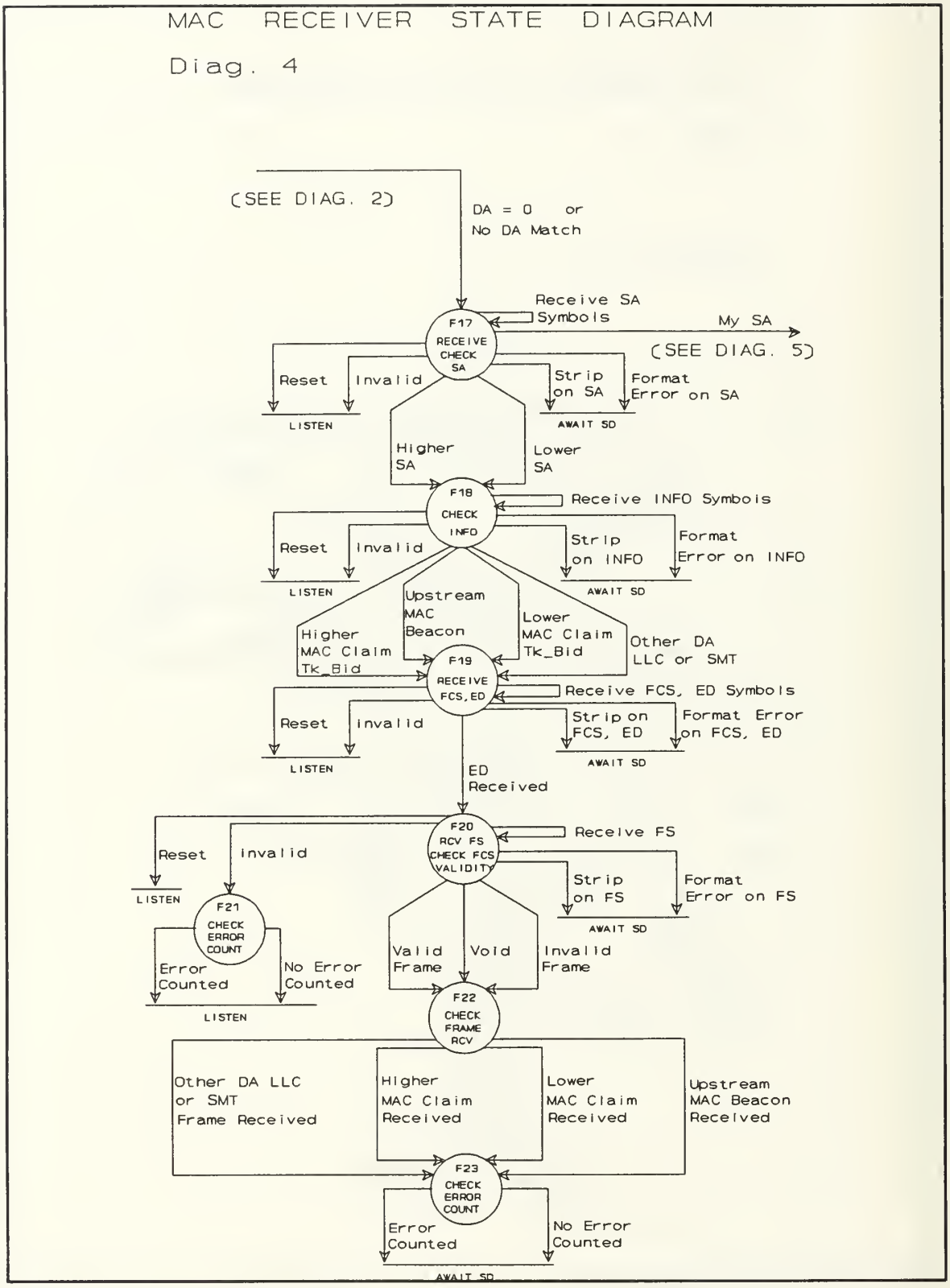


Figure 30: Receiver Diagram 4

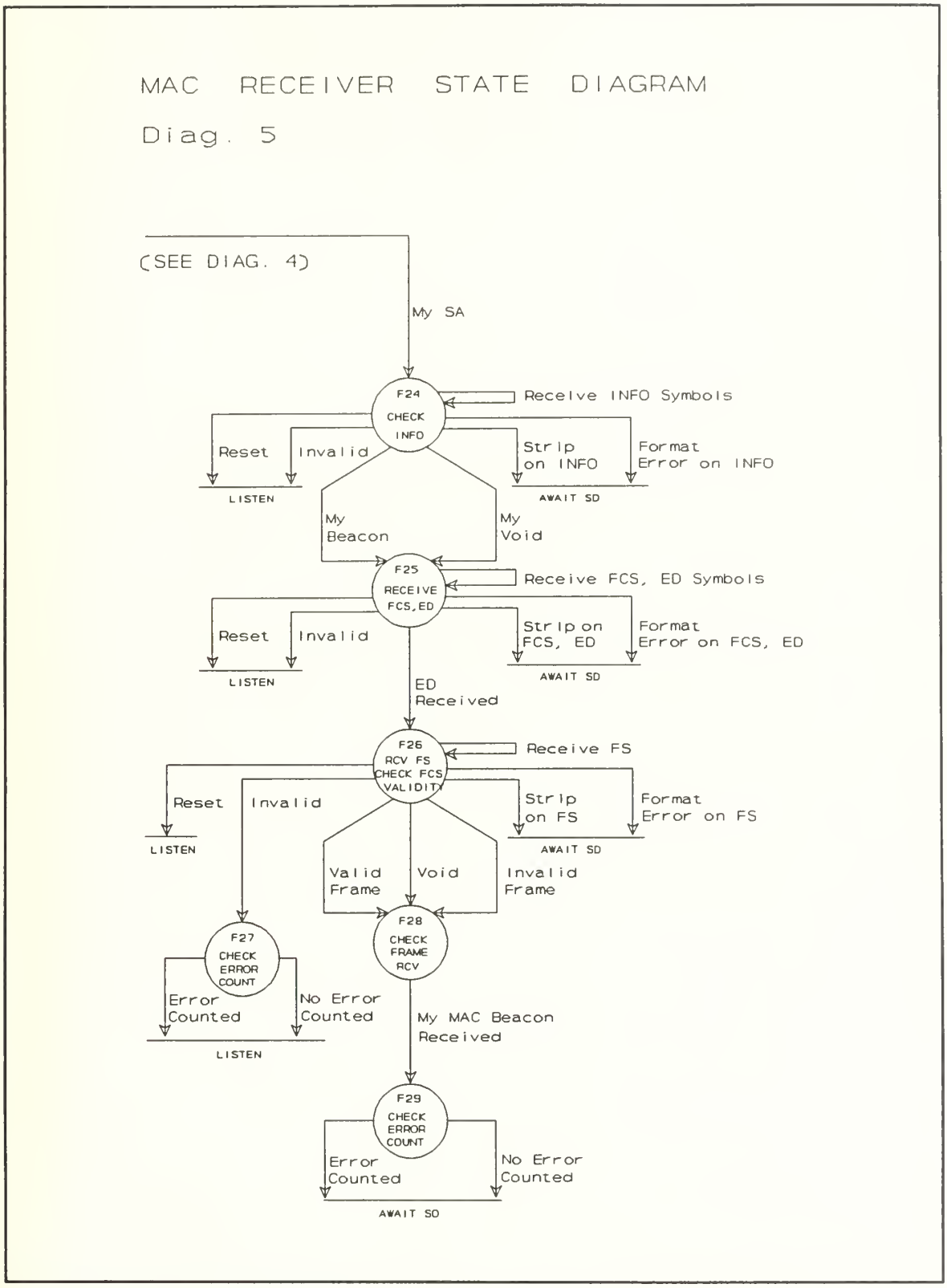


Figure 31: Receiver Diagram 5

MAC RECEIVER STATE DIAGRAM

Diag. 6

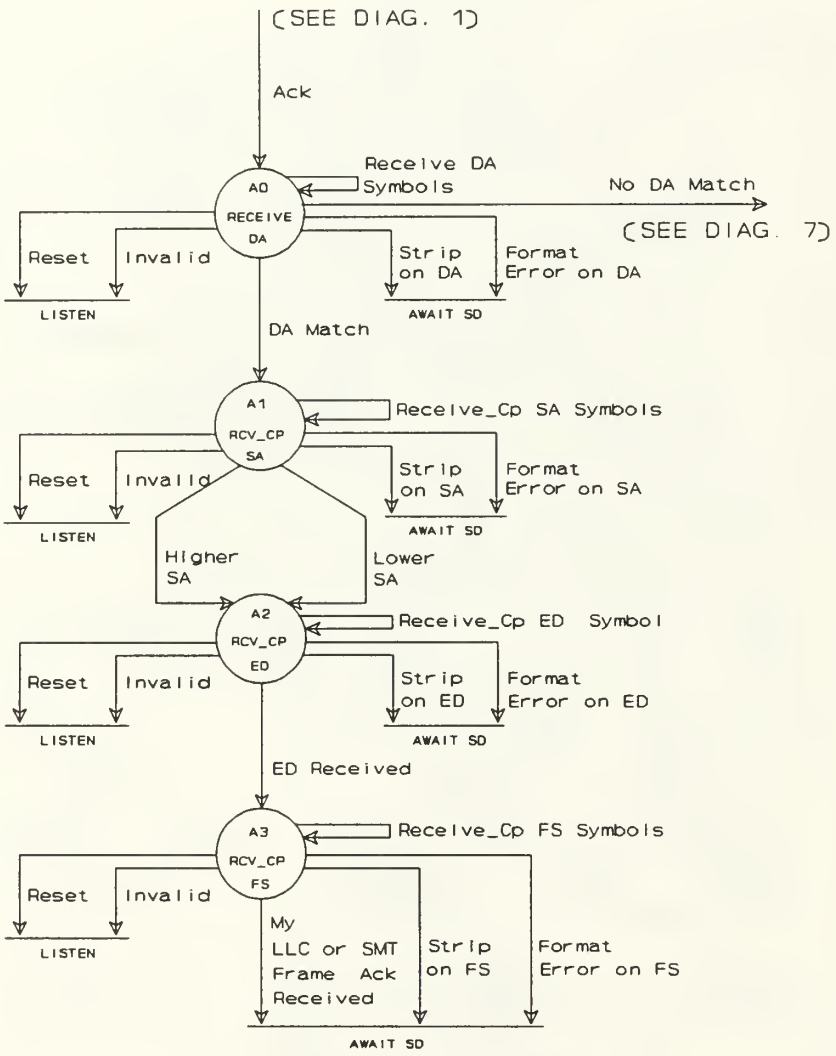


Figure 32: Receiver Diagram 6

MAC RECEIVER STATE DIAGRAM

Diag. 7

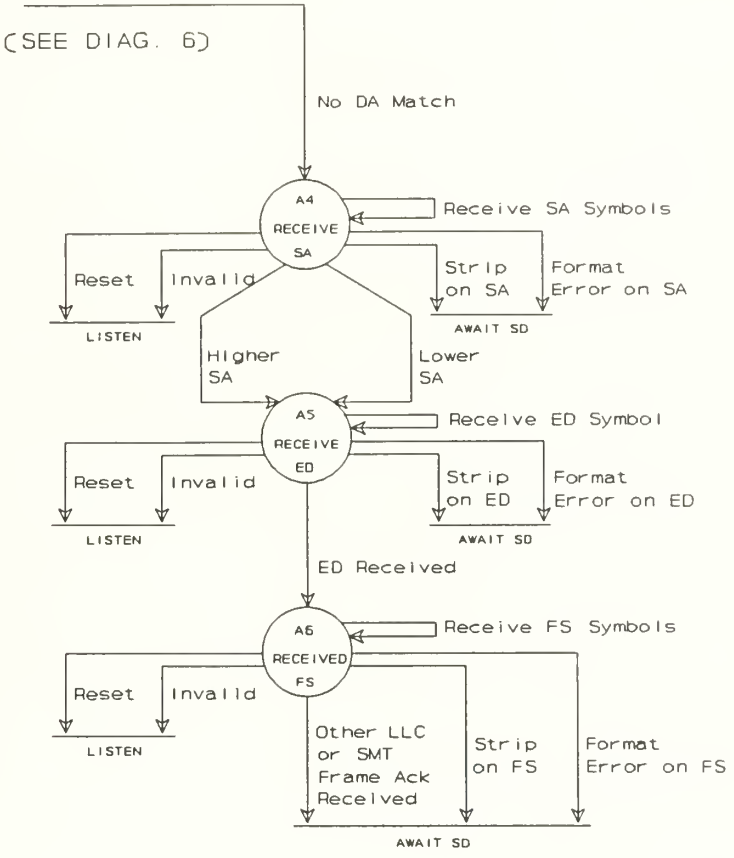


Figure 33: Receiver Diagram 7

MAC RECEIVER STATE DIAGRAM

Diag. 8

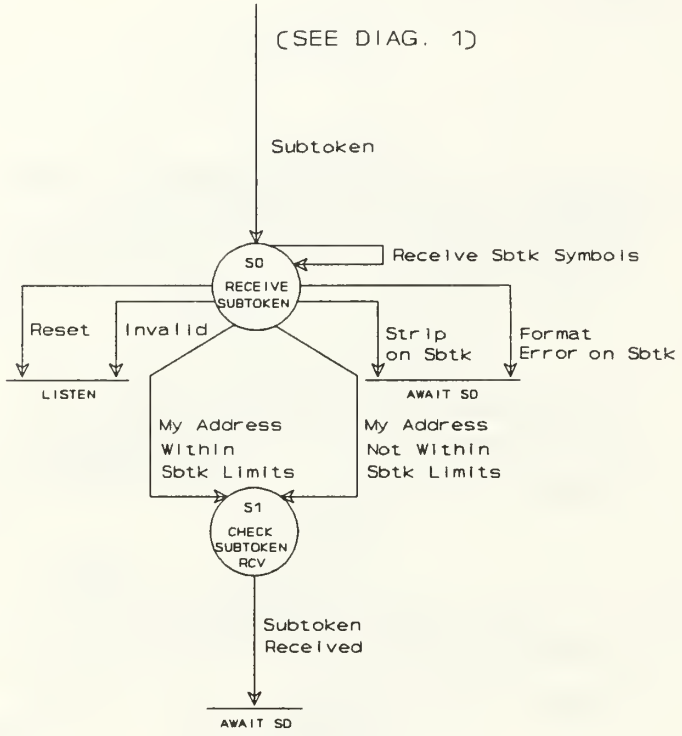


Figure 34: Receiver Diagram 8

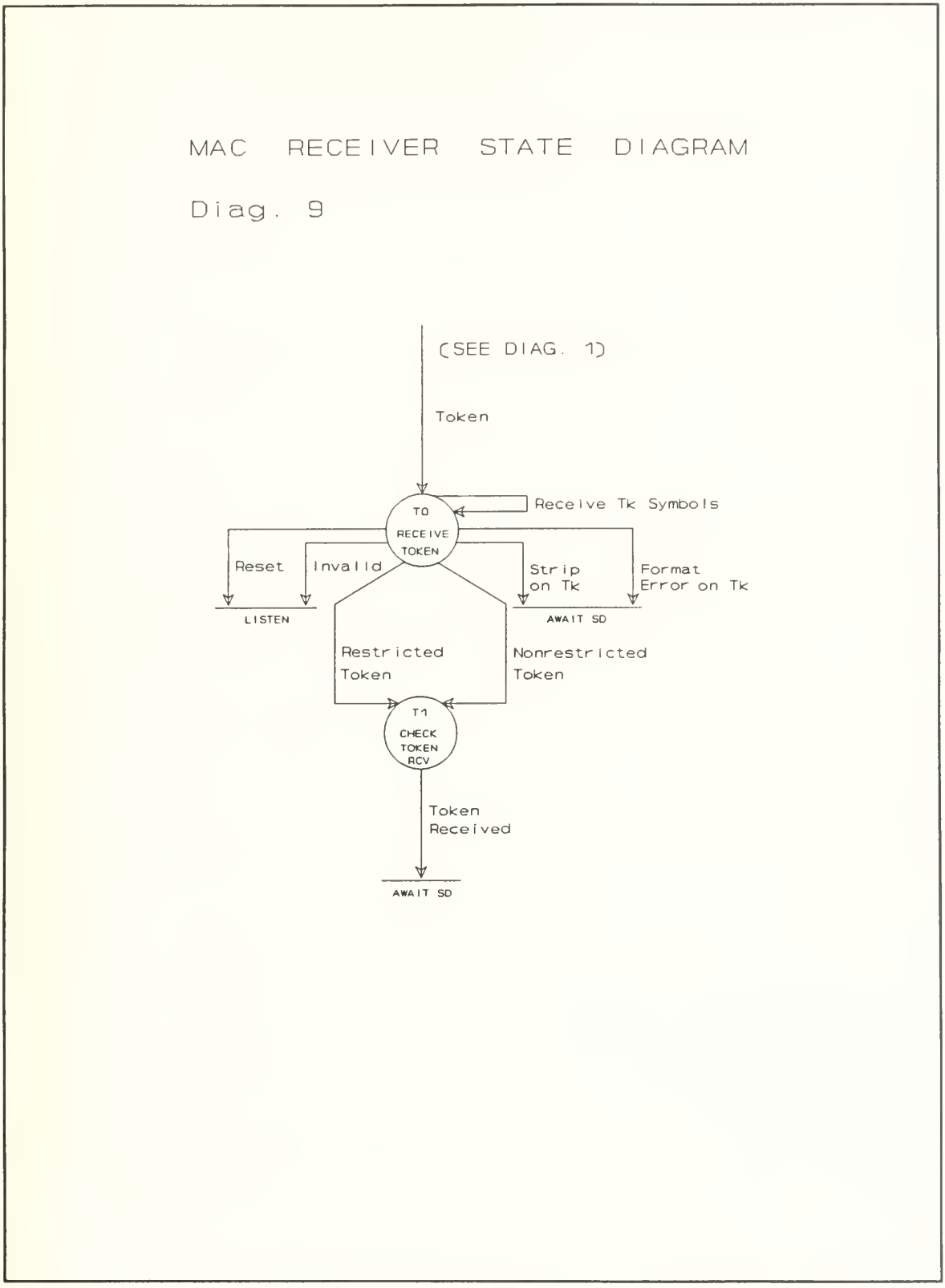


Figure 35: Receiver Diagram 9

Table 3 is the MAC Receiver Transition Table. The table contains all the transitions showed in the diagram. Transitions are represented by short words. Same transitions that occur in different protocol modules are expressed by the combination of the letter-number of the outgoing state and the letter-number of the entering state. Transitions in which symbols are being received show the enabling predicate with the current contents of the PH_indication buffer and the current position of the "symbol" pointer on the array of symbols of that buffer. All boolean variables when stated solely by their names on the enabling predicate are assumed to be true unless otherwise specified.

TABLE 3: MAC RECEIVER TRANSITION TABLE

Transition	Enabling Predicate	Action
Reset R(r,0); F(f,0); A(a,0); S(0,0); T(0,0); r=0,1,2,3; f=0,2,3,4,5, 11,12,13, 17,18,19,20, 24,25,26; a=0,1,2,3,4,5 ,6;	MAC_Reset	T_Neg ← T_Max;
Signal Start R(0,1);	PH_Indication(symbol) = { $PA_r[I_1]$ } ∧ (symbol = $PA_r[I_1]$)	TVX ← reset; TVX ← enabled; symbol ← 0; symbol ← symbol + 1; SIGNAL Idle;
Invalid R(1,0);	PH_Invalid	-
Receive Next Idle Symbol R(1-1);	PH_Indication(symbol) = { $PA_r[I_1]$, $PA_r[I_2..I_{max}]$ } ∧ (symbol = $PA_r[I_i]$ ∋ $1<i\leq max$)	symbol ← symbol + 1;

Transition	Enabling Predicate	Action
Format Error on SD R(1-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}] \wedge \neg SD_r[J]$ }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0;
Start R(1-2);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J]$ }	Idle ← off; SIGNAL Rc_Start; A_, C_, E_, N_, H_, L_, M_Flag ← clear; symbol ← symbol + 1; symbol_ct ← 1;
Invalid R(r,0); F(f,0); A(a,0); S(0,0); T(0,0); r=0,1,2,3; f=0,2,3,4, 11,12,17,18,19,24,25; a=0,1,2,3,4,5,6;	PH_Invalid	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1;
Receive Next SD Symbol R(2-2);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$ } $\wedge$ (symbol = $SD_r[K]$)	symbol ← symbol + 1;
Strip on SD R(2-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $(SD[J \wedge \neg K]) \vee (SD[J] \wedge PA_r[I_1])$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
SFS Received R(2-3);	PH_Indication(symbol) = { Start of Frame Sequence = $PA_r[I_1..I_{max}]$, $SD_r[J,K]$ }	
Receive FC Symbols R(3-3);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$ } $\wedge$ (symbol = $FC_r[n_i]$ ∋ $1 \le i \le 2$)	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Frame R3-F0;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$ } $\wedge$ (FC_r = Frame)	SIGNAL PDU_Frame
Ack R3-A0;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$ } $\wedge$ (FC_r = Acknowledgment)	SIGNAL PDU_Ack
Strip on FC R(3-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], $PA_r[I_1]$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;

Transition	Enabling Predicate	Action
Format Error on FC R(3-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, ($\neg PA_r[I] \vee \neg FC_r[n,n]$) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
Token R3-T0;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$ } ∧ (FC_r = Token)	SIGNAL PDU_Tk
Subtoken R3-S0;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$ } ∧ (FC_r = Subtoken)	SIGNAL PDU_SbTk
Receive DA symbols F(0-0); A(0-0);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$ } ∧ (symbol = $DA_r[n_i]$ ∋ $1 \le i \le 12$)	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
DA Match F0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$ } ∧ [(FC_r.L = 0) ∧ ($DA_r \in$ {SSA})] ∨ [(FC_r.L = 1) ∧ ($DA_r \in$ {SLA})]	A_Flag ← set;
Strip on DA F0-R1; A0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD[J,K]$, $FC_r[n,n]$, $PA_r[I_1]$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on DA F0-R1; A0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, ($\neg PA_r[I] \vee \neg DA_r[4n \vee 12n]$) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
DA null or no DA match F(0-17);	(DA_r = null) ∨ ($DA_r \notin$ {SSA}) ∨ ($DA_r \notin$ {SLA}	
Copy Frame to Local Entity (LLC, SMT, MAC) F(1-2);	A_Flag	Copy_Frame ← true; SIGNAL FR_Strip;
Set Next Station Addressing Flag F(1-2);	A_Flag ∧ (FC_r = Next Station Addressing Frame)	N_Flag ← set

Transition	Enabling Predicate	Action
Receive_Cp SA Symbols F(2-2); A(1-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$ } $\wedge$ Copy_Frame $\wedge$ (symbol = $SA_r[n_i] \ni 1 \leq i \leq 12$)	Rcv-buf(symbol) $\leftarrow$ PH_Indication(symbol) ; symbol $\leftarrow$ symbol + 1; symbol_ct $\leftarrow$ symbol_ct + 1;
Higher SA F(2-3); F(17-18); A(1-2); A(4-5);	[(FC_r.L = 0) $\wedge$ (SA_r > MSA) $\wedge$ MLA = 0] $\wedge$ [(FC_r.L = 1) $\wedge$ (SA_r = MLA)]	H_Flag $\leftarrow$ set;
Lower SA F(2-3); F(17-18); A(1-2); A(4-5);	[(SA_r > 0) $\wedge$ ($FC.L_r$ = 0) $\wedge$ (SA_r < MSA) $\wedge$ (MLA = 0)] $\vee$ [(FC_r.L = 1) $\wedge$ (SA_r < MLA)]	L_Flag $\leftarrow$ set;
My SA F(2-11);	[($FC.L_r$ = 0) $\wedge$ (SA_r = MSA) $\ni$ MSA > 0] $\vee$ [($FC.L_r$ = 1) $\wedge$ (SA_r = MLA) $\ni$ MLA > 0]	M_Flag $\leftarrow$ set;
Strip on SA F(f)-R1; A1-R1; f=2,17;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $PA_r[I_1]$ }	SIGNAL Idle; symbol $\leftarrow$ 0; symbol $\leftarrow$ symbol + 1; symbol_ct $\leftarrow$ 0;
Format Error on SA F(f)-R1; A1-R1; f=2,17;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, ($\neg PA_r[I] \vee \neg SA_r[4n \vee 12n]$) }	SIGNAL FO_Error; Lost_Ct $\leftarrow$ Lost_Ct + 1; symbol $\leftarrow$ 0; symbol_ct $\leftarrow$ 0;
Receive_Cp INFO symbols F(3-3); F(13-13);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$ } $\wedge$ Copy_Frame $\wedge$ (symbol = $INFO_r[n_i] \ni 1 < i \leq max$)	Rcv-buf(symbol) $\leftarrow$ PH_Indication(symbol) ; symbol $\leftarrow$ symbol + 1; symbol_ct $\leftarrow$ symbol_ct + 1;
Strip on INFO F(f)-R1; f=3,11,18, 24;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $PA_r[I_1]$ }	SIGNAL Idle; symbol $\leftarrow$ 0; symbol $\leftarrow$ symbol + 1; symbol_ct $\leftarrow$ 0;
Format Error on INFO F(f)-R1; f=3,11,18, 24;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, ($\neg PA_r[I] \vee \neg INFO_r[n_1..n_{max}]$ }	SIGNAL FO_Error; Lost_Ct $\leftarrow$ Lost_Ct + 1; symbol $\leftarrow$ 0; symbol_ct $\leftarrow$ 0;

Transition	Enabling Predicate	Action
DA=MA LLC or SMT Frame with Sbtk F(3-4);	(FC_r = LLC Frame with subtoken ∨ FC_r = SMT Frame with subtoken) ∧ A_Flag ∧ ¬E_Flag ∧ (L_Flag ∨ H_Flag)	SIGNAL My_DA_LLC_or_ SMT_with_Sbtk;
DA=MA LLC or SMT Frame without Sbtk F(3-4);	(FC_r = LLC Frame without subtoken ∨ FC_r = SMT Frame without subtoken) ∧ A_Flag ∧ ¬E_Flag ∧ (L_Flag ∨ H_Flag)	SIGNAL My_DA_LLC_or_ SMT_without_Sbtk;
Strip on FCS, ED F(f)-R1; f=4,12,19, 25;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $PA_r[I_1]$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on FCS, ED F(f)-R1; f=4,12,19, 25;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, ($\neg PA_r[I] \vee \neg FCS_r[8n] \vee \neg ED_r[T]$) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
ED Received_ Cp F(4-5); F(12-13);	Rcv-buf(symbol), PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$ }	Frame_Ct←Frame_Ct + 1;
Receive Determine FS Symbols F(5-5);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } ∧ Copy_Frame ∧ (symbol = ($FS.E_r[S/R]$ ∧ $FS.A_r[S/R]$ ∧ $FS.C_r[S/R]$))	Rcv-buf(symbol) ← PH_Indication(symbol); symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Invalid F(5-6); F(13-14); F(20-21); F(26-27);	PH_Invalid	SIGNAL FR_Received; Copy_Frame ← false; Fr_Strip ← off;
Valid Frame F(5-7); F(13-15); F(20-22); F(26-28);	[($FS.E_r$ = R) ∧ (Valid Data Length)] ∨ [(Valid FCS_r) ∧ (FC_r.FF = Implementor)]	TVX ← Reset; E_Flag ← clear; Valid_Frame ← true;

Transition	Enabling Predicate	Action
Void F(5-7); F(13-15); F(20-22); F(26-28);	[(FC_r = Void) ∧ ($FS.E_r$ = R) ∧ ¬[(Valid Data Length) ∨ (Valid FCS_r)]	E_Flag ← set; A_, M_, H_, L_, N_Flag ← clear; Valid_Frame ← false;
Invalid Frame F(5-7); F(13-15); F(20-22); F(26-28);	[($FS.E_r$ ≠ R) ∨ ¬(Valid Data Length)] ∨ ¬[(Valid FCS_r) ∨ (FC_r.FF = Implementor)]	E_Flag ← set; A_, M_, H_, L_, N_Flag ← clear; Valid_Frame ← false;
Strip on FS F(f)-R1; f=5,13,20, 26;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], FC_r[n,n], DA_r[4n ∨ 12n], SA_r[4n ∨ 12n], $INFO_r[n_1..n_{max}]$, FCS_r[8n], ED_r[T], $PA_n[I_1]$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on FS F(f)-R1; f=5,13,20, 26;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD_r[J,K], FC_r[n,n], DA_r[4n ∨ 12n], SA_r[4n ∨ 12n], $INFO_r[n_1..n_{max}]$, FCS_r[8n], ED_r[T] ($\neg PA_n[I]$ ∨ $\neg(FS.E_n[S/R]$, ∨ $FS.A_n[S/R]$, ∨ $FS.C_n[S/R])$) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
Error Counted $F(f_1)$-R0; $F(f_2)$-R1; f_1=6,14,21, 27; f_2=10,16,23,29;	(E_Flag) ∧ (E_r ≠ S)	Error_Ct ← Error_Ct + 1;
No Error Counted $F(f_1)$-R0; $F(f_2)$-R1; f_1=6,14,21, 27; f_2=10,16,23,29;	¬(E_Flag) ∨ (E_r = S)	-
Frame Copied F(7-8);	Valid_Frame ∧ Copy_Frame	C_Flag ← Set;
Frame Not Copied F(7-8);	Valid_Frame = false ∨ Copy_Frame = false	C_Flag ← clear;

Transition	Enabling Predicate	Action
Ack Frame F(8-9);	$FS.A_r = R \wedge C\_Flag$	N_Flag ← clear; SIGNAL Ack_Frame;
No Ack Frame F(8-9);	$FS.A_r \neq R \wedge N\_Flag \vee \neg(C\_Flag)$	Ack_Frame ← off;
Suspected DA Received F(8-9);	$(FS.A_r = S) \wedge (A\_Flag) \wedge (DA_r.IG = 0) \wedge \neg(E\_Flag) \wedge (FC_r.FF = 0)$	Notify_SMT(suspected DA_received)
LLC or SMT Frame with DA=MA Received F(9-10);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (FC_r = LLC Frame $\vee$ FC_r = SMT Frame) $\wedge$ $\neg(E\_Flag) \wedge A\_Flag$	R_Flag ← clear; SIGNAL FR_Received; Copy_Frame ← false; Fr_Strip ← off;
My Void F(11-12); F(24-25);	(FC_r = Void) $\wedge$ (A_Flag) $\wedge$ (M_Flag)	TVX ← reset; SIGNAL My_Void;
My MAC Claim Token Bid F(11-12);	(FC_r = Claim) $\wedge$ (A_Flag) $\wedge$ (M_Flag) $\wedge$ ($T\_Max \leq T\_Bid\_Rc \leq T\_Min$) $\wedge$ ($T\_Bid\_Rc = T\_Req$)	SIGNAL My_Claim; T_Neg ← T_Bid_Rc;
Receive_Cp FCS, ED Symbols F(12-12);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, *$FCS_r[8n]$, $ED_r[T]$* } $\wedge$ Copy_Frame $\wedge$ (symbol = $FCS_r[n_i]$ $\ni$ $1<i\leq 8$)	Rcv-buf(symbol) ← PH_Indication(symbol); symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Receive_Cp FS Symbols F(13-13);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, *$FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$* } $\wedge$ Copy_Frame $\wedge$ (symbol = ($FS.E_r[S/R] \wedge FS.A_r[S/R] \wedge FS.C_r[S/R]$))	Rcv-buf(symbol) ← PH_Indication(symbol); symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
My MAC Claim Received F(15-16);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (Valid_Frame) $\wedge$ (My_Claim)	T_Neg ← T_Bid_Rc; SIGNAL FR_Received; Copy_Frame ← false; Fr_Strip ← off; My_Claim ← off;
Receive SA Symbols F(17-17); A(4-4);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, *$SA_r[4n \vee 12n]$* } $\wedge$ (symbol = $SA_r[n_i]$ $\ni$ $1\leq i\leq 12$	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;

Transition	Enabling Predicate	Action
My SA F(17-24);	[(FC.L$_r$ = 0) ∧ (SA$_r$ = MSA) ∋ MSA > 0] ∨ [(FC.L$_r$ = 1) ∧ (SA$_r$ = MLA) ∋ MLA > 0]	SIGNAL Fr_Strip; M_Flag ← set;
Receive INFO symbols F(18-18); F(24-24);	PH_Indication(symbol) = { PA$_r$[I$_1$..I$_{max}$], SD$_r$[J,K], FC$_r$[n,n], DA$_r$[4n ∨ 12n], SA$_r$[4n ∨ 12n], *INFO$_r$[n$_1$..n$_{max}$]* }	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Higher MAC Claim Tk Bid F(18-19);	(FC$_r$ = Claim) ∧ (H_Flag) ∧ (T_Max ≤ T_Bid_Rc ≤ T_Min) ∧ (T_Bid_Rc > T_Req)	SIGNAL Higher_Claim; T_Neg ← T_Bid_Rc;
Upstream MAC Beacon F(18-19);	(FC$_r$ = Beacon) ∧ ¬(M_Flag)	SIGNAL Other_Beacon; T_Neg ← T_Max;
Lower MAC Claim Tk Bid F(18-19);	(FC$_r$ = Claim) ∧ ¬[(H_Flag) ∨ (M_Flag)] ∧ {[(MSA enabled) ∋ (MSA > 0)] ∨ [(MLA enabled) ∋ (MLA > 0)]} ∧ (T_Max ≤ T_Bid_Rc ≤ T_Min) ∧ (T_Bid_Rc < T_Req)	SIGNAL Lower_Claim; T_Neg ← T_Bid_Rc;
Other DA LLC or SMT Frame F(18-19);	(FC$_r$ = LLC Frame ∨ FC$_r$ = SMT Frame) ∧ ¬(E_Flag) ∧ (L_Flag ∨ H_Flag)	SIGNAL Other_LLC_or_SMT;
ED Received F(19-20); F(25-26);	PH_Indication(symbol) = {PA$_r$[I$_1$..I$_{max}$], SD$_r$[J,K], FC$_r$[n,n], DA$_r$[4n ∨ 12n], SA$_r$[4n ∨ 12n], INFO$_r$[n$_1$..n$_{max}$], FCS$_r$[8n], ED$_r$[T]}	Frame_Ct←Frame_Ct + 1;
Receive FS Symbols F(20-20); F(20-26);	PH_Indication(symbol) = { PA$_r$[I$_1$..I$_{max}$], SD$_r$[J,K], FC$_r$[n,n], DA$_r$[4n ∨ 12n], SA$_r$[4n ∨ 12n], INFO$_r$[n$_1$..n$_{max}$], FCS$_r$[8n], ED$_r$[T], *FS.E$_r$[S/R], FS.A$_r$[S/R], FS.C$_r$[S/R]* } ∧ (symbol = (FS.E$_r$[S/R] ∧ FS.A$_r$[S/R] ∧ FS.C$_r$[S/R]))	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Other DA LLC or SMT Frame Received F(22-23);	PH_Indication(symbol) = { PA$_r$[I$_1$..I$_{max}$], SD$_r$[J,K], FC$_r$[n,n], DA$_r$[4n ∨ 12n], SA$_r$[4n ∨ 12n], INFO$_r$[n$_1$..n$_{max}$], FCS$_r$[8n], ED$_r$[T], FS.E$_r$[S/R], FS.A$_r$[S/R], FS.C$_r$[S/R] } ∧ (Valid_Frame) ∧ (Other_LLC_or_SMT)	SIGNAL FR_Received; Other_LLC_or_SMT ← off;

Transition	Enabling Predicate	Action
Higher MAC Claim Received F(22-23);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (Valid_Frame) $\wedge$ (Higher_Claim)	SIGNAL FR_Received; Higher_Claim ← off;
Lower MAC Claim Received F(22-23);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (Valid_Frame) $\wedge$ (Lower_Claim)	SIGNAL FR_Received; Lower_Claim ← off;
Upstream MAC Beacon Received F(22-23);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (Valid_Frame) $\wedge$ (Other_Beacon)	SIGNAL FR_Received; Other_Beacon ← off;
My Beacon F(24-25);	(FC_r = Beacon) $\wedge$ (M_Flag)	SIGNAL My_Beacon; T_Neg ← T_Max;
My MAC Beacon Received F(28-29);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (Valid_Frame) $\wedge$ (My_Beacon)	SIGNAL FR_Received; My_Beacon ← off;
DA Match A0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$ } $\wedge$ [(FC_r.L = 0) $\wedge$ ($DA_r \in$ {SSA})] $\vee$ [(FC_r.L = 1) $\wedge$ ($DA_r \in$ {SLA})]	A_Flag ← set; Copy_ack ← true; SIGNAL FR_Strip;
Receive_Cp ED Symbol A(2-2);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$ } $\wedge$ Copy_Frame $\wedge$ (symbol = $ED_r[T]$)	Rcv-buf(symbol) ← PH_Indication(symbol); symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Strip on ED A(2-1); A(5-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $PA_r[I_1]$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on ED A(2-1); A(5-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, ($\neg PA_r[I] \vee \neg ED_r[T]$) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;

Transition	Enabling Predicate	Action
ED Received_ Cp A(2-3);	Rcv-buf(symbol), PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$ }	Ack_Ct ← Ack_Ct + 1;
Receive_Cp FS Symbols A(3-3);	Rcv-buf(symbol), PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$, *$FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$* } ∧ Copy_Ack ∧ (symbol = ($FS.E_r[S/R] \wedge FS.A_r[S/R] \wedge FS.C_r[S/R]$))	Rcv-buf(symbol) ← PH_Indication(symbol); symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
My LLC or SMT Frame Ack Received A(3-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } ∧ (FC_r = LLC_Frame_Ack ∨ FC_r = SMT_Frame_ack) ∧ M_Flag	SIGNAL Ack_Received; Copy_Ack ← false; FR_Strip ← off;
Strip on FS A(a)-R1; a=3,6;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$, *$PA_r[I_1]$* }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on FS A(a)-R1; a=3,6;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$ (*$\neg PA_r[I] \vee \neg(FS.E_r[S/R], \vee FS.A_r[S/R], \vee FS.C_r[S/R])$*) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
no DA match A(0-4);	(DA_r = null) ∨ ($DA_r \notin$ {SSA}) ∨ ($DA_r \notin$ {SLA}	
Receive ED Symbol A(5-5);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, *$ED_r[T]$* } ∧ (symbol = $ED_r[T]$)	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
ED Received A(5-6);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$ }	Ack_Ct ← Ack_Ct+1;
Receive FS Symbols A(6-6);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$, *$FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$* } ∧ Copy_Ack ∧ (symbol = ($FS.E_r[S/R] \wedge FS.A_r[S/R] \wedge FS.C_r[S/R]$))	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;

Transition	Enabling Predicate	Action
Other LLC or SMT Frame Ack Received A(3-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (FC_r = LLC_Frame_Ack $\vee$ FC_r = SMT_Frame_ack) $\wedge$ $\neg$(M_Flag)	SIGNAL Ack_Received;
Receive Sbtk Symbols S(0-0);	PH_Indication(symbol) = { $PA_r[I_1..I_n]$, $SD_r[J,K]$, FC_r[CLFF,ZZZZ], *SL_r[4n $\vee$ 12n], EL_r[4n or 12n], $CLASS_r$[2n], ED_r[T,T]* } $\wedge$ (symbol = $SL_r[n_i] \ni 1 \le i \le 12$)	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Strip on Sbtk S0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, *$PA_r[I_1]$* } $\vee$ { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $SL_r[4n \vee 12n]$, *$PA_r[I_1]$* } $\vee$ { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $SL_r[4n \vee 12n]$, EL_r[4n or 12n], *$PA_r[I_1]$* } $\vee$ { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $SL_r[4n \vee 12n]$, EL_r[4n or 12n], $CLASS_r$[2n], *$PA_r[I_1]$* }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on Sbtk S0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, ($\neg$*$PA_r[I_1]$* $\vee$ $\neg$*$SL_r[4n \vee 12n]$*) } $\vee$ { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $SL_r[4n \vee 12n]$, ($\neg$*$PA_r[I_1]$* $\vee$ $\neg$*$EL_r[4n \vee 12n]$*) } $\vee$ { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $SL_r[4n \vee 12n]$, EL_r[4n or 12n], ($\neg$*$PA_r[I_1]$* $\vee$ $\neg$*$CLASS_r[2n]$*) } $\vee$ { $PA_r[I_1..I_{max}]$, SD[J,K], $FC_r[n,n]$, $SL_r[4n \vee 12n]$, EL_r[4n or 12n], $CLASS_r$[2n], ($\neg$*$PA_r[I_1]$* $\vee$ $\neg$*$ED_r[T,T]$*) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
My Address Within Sbtk Limits S(0-1);	$SL_r \le MA \le EL_r$	SbTk_Flag ← Set; Write SbTk_Class;
My Address not Within SbTk Limits S(0-1);	($MA < SL_r$) $\vee$ ($MA > EL_r$) $\vee$ ($SL_r < EL_r$)	SbTk_Flag ← Reset; TVX ← Reset;
Subtoken Received S1-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_n]$, $SD_r[J,K]$, FC_r[CLFF,ZZZZ], $SL_r[4n \vee 12n]$, EL_r[4n or 12n], $CLASS_r$[2n], ED_r[T,T] }	SIGNAL SbTk_Received;

Transition	Enabling Predicate	Action
Receive Tk Symbols T(0-0);	PH_Indication(symbol) = { $PA_r[I_1..I_n]$, $SD_r[J,K]$, FC_r[CLFF,ZZZZ], $ED_r[T,T]$ } ∧ (symbol = $ED_r[T_1]$)	symbol ← symbol + 1; symbol_ct ← symbol_ct + 1;
Strip on Tk T0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], FC_r[n,n], $PA_r[I_1]$ }	SIGNAL Idle; symbol ← 0; symbol ← symbol + 1; symbol_ct ← 0;
Format Error on Tk T0-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, SD[J,K], FC_r[n,n], ($\neg PA_r[I_1]$ ∨ $\neg ED_r[T,T]$) }	SIGNAL FO_Error; Lost_Ct ← Lost_Ct + 1; symbol ← 0; symbol_ct ← 0;
Non Restricted Token T(0-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, FC_r[n,n], ED_r[T,T] } ∧ ($FC.C_r$ = Nonrestricted)	R_Flag ← Clear; TVX ← Reset;
Restricted Token T(0-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, FC_r[n,n], ED_r[T,T] } ∧ ($FC.C_r$ = Restricted)	R_Flag ← Set;
Token Received T1-R1;	PH_Indication(symbol) = { $PA_r[I_1..I_n]$, $SD_r[J,K]$, FC_r[n,n], ED_r[T,T] }	SIGNAL Tk_Received;

5. Mac Transmitter Operation and Specification

The Mac transmitter repeats symbols from other stations over the ring after a station delay, and transmits its own symbols which come from the appropriate buffers. The transmitter operates on the input symbol stream from PHY (PH_indication) and produces the output symbol stream to PHY (PH_request). Its operation is synchronized by signals from the Receiver. The Transmitter repeats, removes, and inserts a PDU into the ring during different phases of protocol operation. It is responsible for capturing of a token or subtoken, carrying out the timed token operations, transmitting data and issuing a new token. The transmitter is also responsible for transmission of MAC supervisory frames in the recovery and beacon processes.

In this protocol, the MAC of the token holding station controls the transmission of new frames and token on the dual ring by using a $\mathbf{Wait}_{primary}$ and $\mathbf{Wait}_{secondary}$ variable set by each transmitter on the appropriate ring. When a transmitter is active (i.e., transmitting its data) the **Wait** variable on that ring is set to true. Upon completion of frame transmission this variable is set to false.

a. The MAC Transmitter State Diagram and Transition Table

Figure 36 depicts the overall view of the MAC Transmitter State Diagram. It shows 11 smaller diagrams which are presented as separate figures. The Idle state is shown in Diagram 1 (central rectangle in Figure 36). The Idle state is the initial state and indicates the normal condition of the medium. The transition *Tx Idle Symbols* corresponds to the transmission of PDU Preambles (PA). Several examples on the transmitter protocol operation are given in this subsection.

(1) The DA Station Removes a Frame Being Received - no subtoken issued

A path of transitions for a frame without subtoken received with a DA match is shown. The frame is removed from the ring and no subtoken is issued. The path begins in Diagram 1, goes through Diagrams 2 and 3, and returns to Diagram 1.

- I0 *Tx Idle Symbols* I0 *Start* P0 *Tx Next SD Symbol* P0 *SFS Transmitted* P1 *Frame* F0 *DA Match: Frame Without Subtoken Strip Frame* I0 *Tx Idle Symbols* I0

(2) The DA Station Receives a Frame With Subtoken and Uses the Subtoken

The frame with subtoken is received with a DA match and the station uses the subtoken to transmit one synchronous or asynchronous frame.

- I0 *Tx Idle Symbols* I0 *Start* P0 *Tx Next SD Symbol* P0 *SFS Transmitted* P1 *Frame* F0 *DA Match: Frame With Subtoken* F1 *Usable Subtoken* I0 *Tx Idle Symbols* I0 *Usable Subtoken Received* S2 *Tx Syn/Asy Frame Symbols* S2 *End Syn/Asy* S3 *Station Holding Sbtk Tx Completed* I0 *Tx Idle Symbols* I0

(3) A Station Repeats the Frame Downstream on the Ring

A complete path of transitions for the repeat frame protocol operation. This process is accomplished when there is no DA or SA match. The frame is assumed to be repeated with no error detected.

- I0 *Tx Idle Symbols* I0 *Start* P0 *Tx Next SD Symbol* P0 *SFS Transmitted* P1 *Frame* F0 *DA=0 or No DA Match F2 No SA Match: Pass Frame* F3 *Repeat FC,DA,SA,INFO,FCS,ED Symbols* F3 *ED Transmitted* F4 *Reset E* F5 *Tx FS Symbols* F5 *Frame Repeated* I0 *Tx Idle Symbols* I0

(4) Reliability Maintained When Configuration Changes (THRU to WRAP)

Assume that (1) the network configuration changes to the "Wrap mode" (one logical ring) due to a serious ring failure, and (2) the frame is received with a SA Match. The frame is removed from the ring by the SA address station as in the original FDDI protocol. The improved protocol reverts to the basic FDDI operation.

- I0 *Tx Idle Symbols* I0 *Start* P0 *Tx Next SD Symbol* P0 *SFS Transmitted* P1 *Frame* F0 *No DA Match: Wrap Mode* F2 *SA Match: Wrap Mode, Strip Frame* I0 *Tx Idle Symbols* I0

(5) A Station Captures a Token, Performs Simultaneous Transmission, and Issues a new Token.

A complete path of transitions is given for (1) capturing of a token, (2) simultaneous transmission of one synchronous frame on both rings, and (3) after transmission of both frames the token is issued. The frame on the primary ring goes with the subtoken information.

- (1) I0 *Tx Idle Symbols* I0 *Start* P0 *Tx Next SD Symbol* P0 *SFS Transmitted* P1 *Token T0 Ring Opr and Token Early* T1 *Nonrestricted* T2 *Usable Token* T3 *Capture Token* I0 *Tx Idle Symbols* I0 (2)

Tx Syn Frame Symbols$_{primary}$ E0 (a)

↗

(2) I0 *Usable Token Received* E0

↘

Tx Syn Frame Symbols$_{secondary}$ E0 (b)

(a) E0 *Syn Frame With Sbtk Info Issued* E1 *Station Holding Tk Tx Primary Ring Completed* E9 *Ring Opr E10 Nonrestricted* E11 *Wait Other Ring* E11 .....(3) *Issue Token E12 Tx Token Symbols* E12 *token issued I0 Tx Idle Symbols* I0

(b).....E0 *Syn Frame Without Sbtk Issued* E1 *Station Holding Tk Tx Secondary Ring Completed* I0 *Tx Idle Symbols* I0

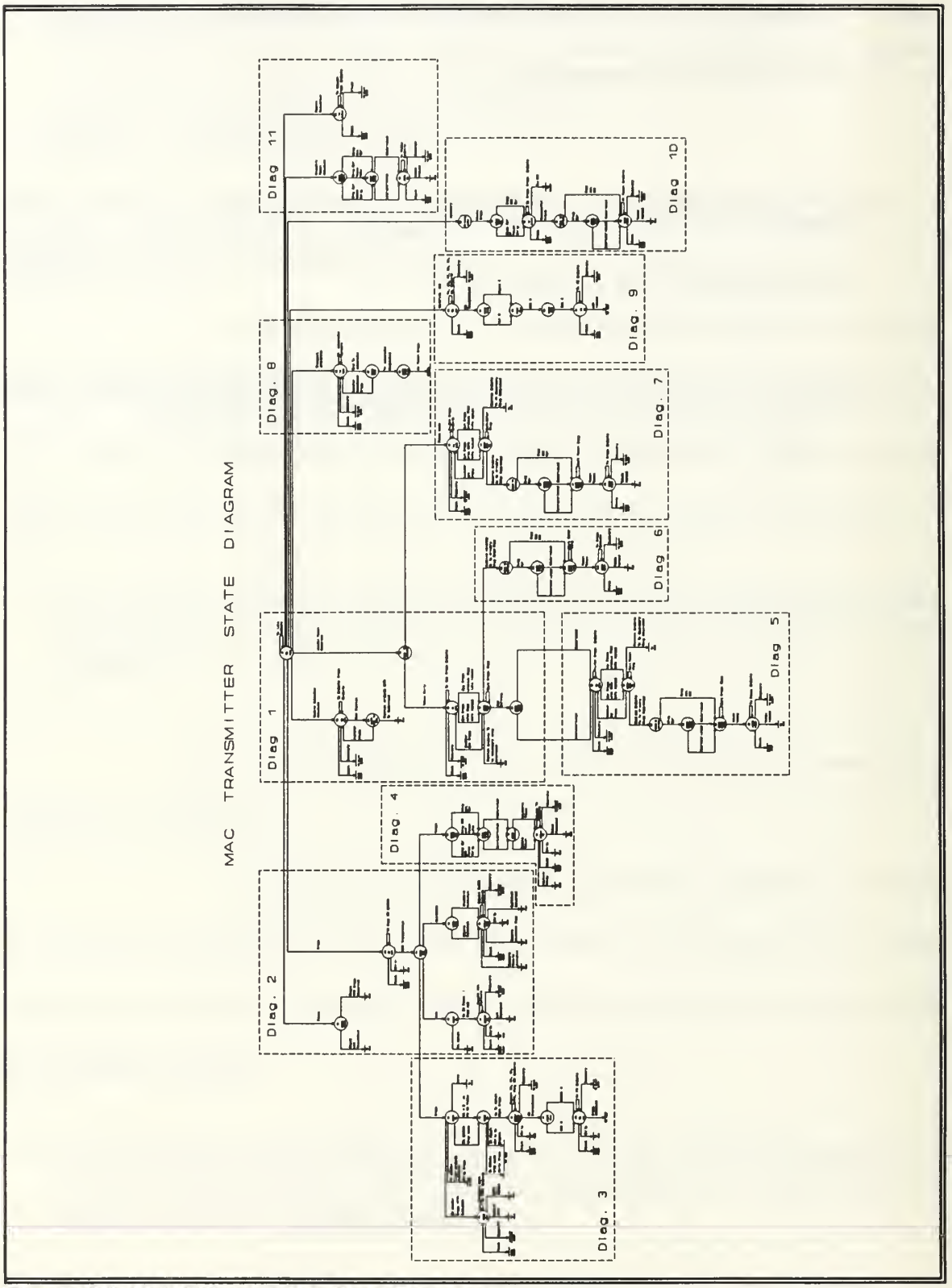


Figure 36: MAC Transmitter State Diagram

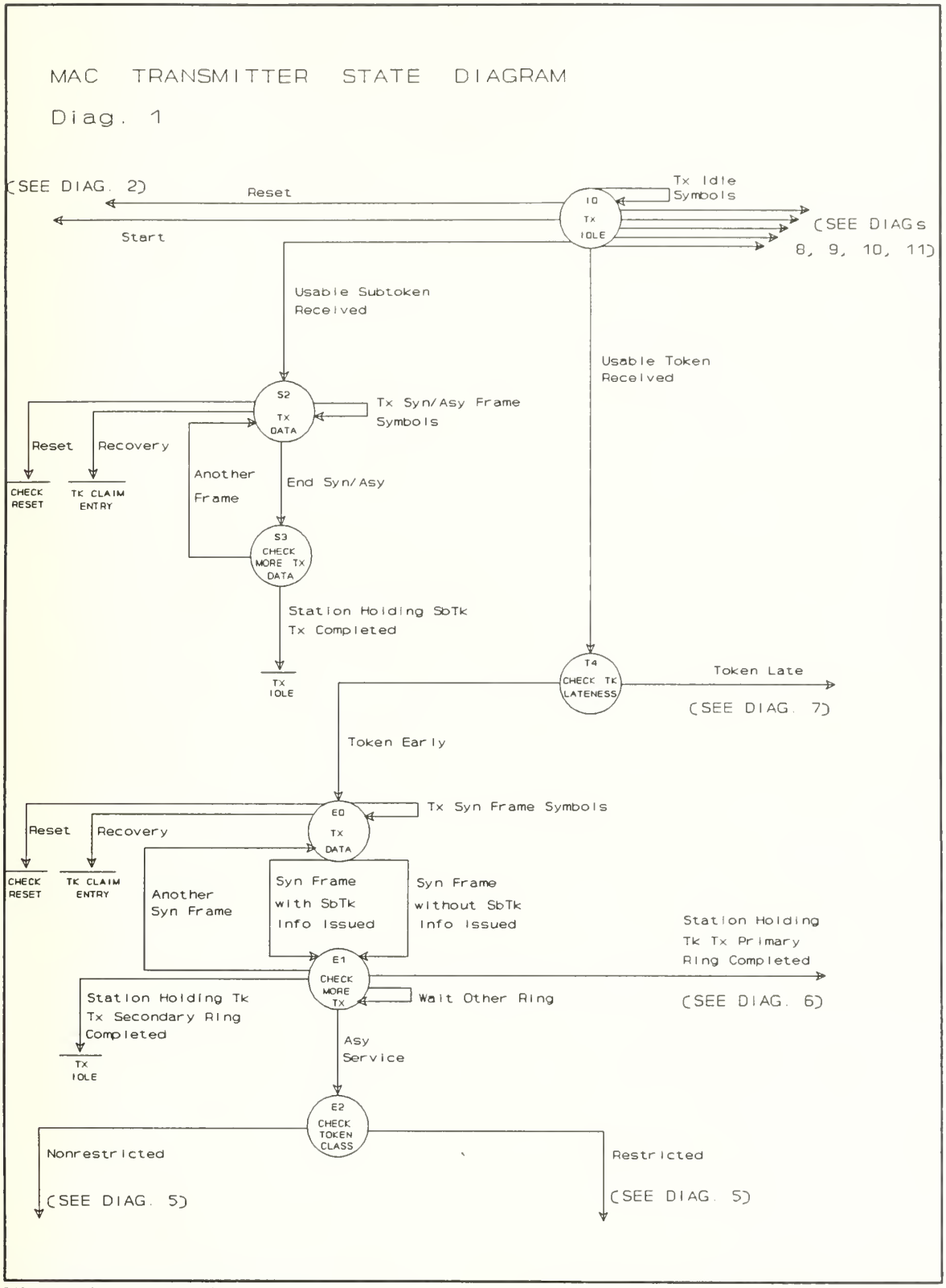


Figure 37: Transmitter Diagram 1

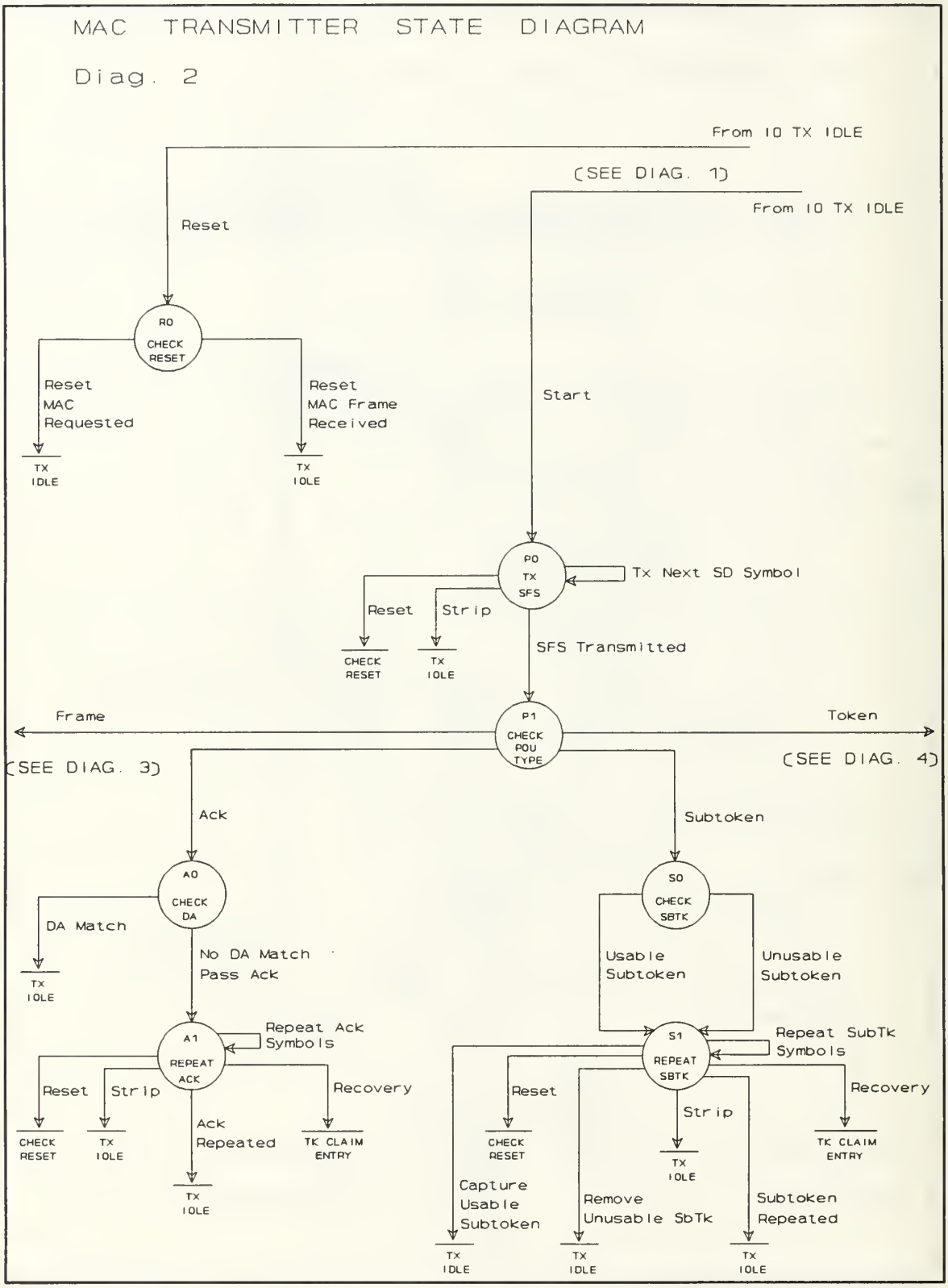


Figure 38: Transmitter Diagram 2

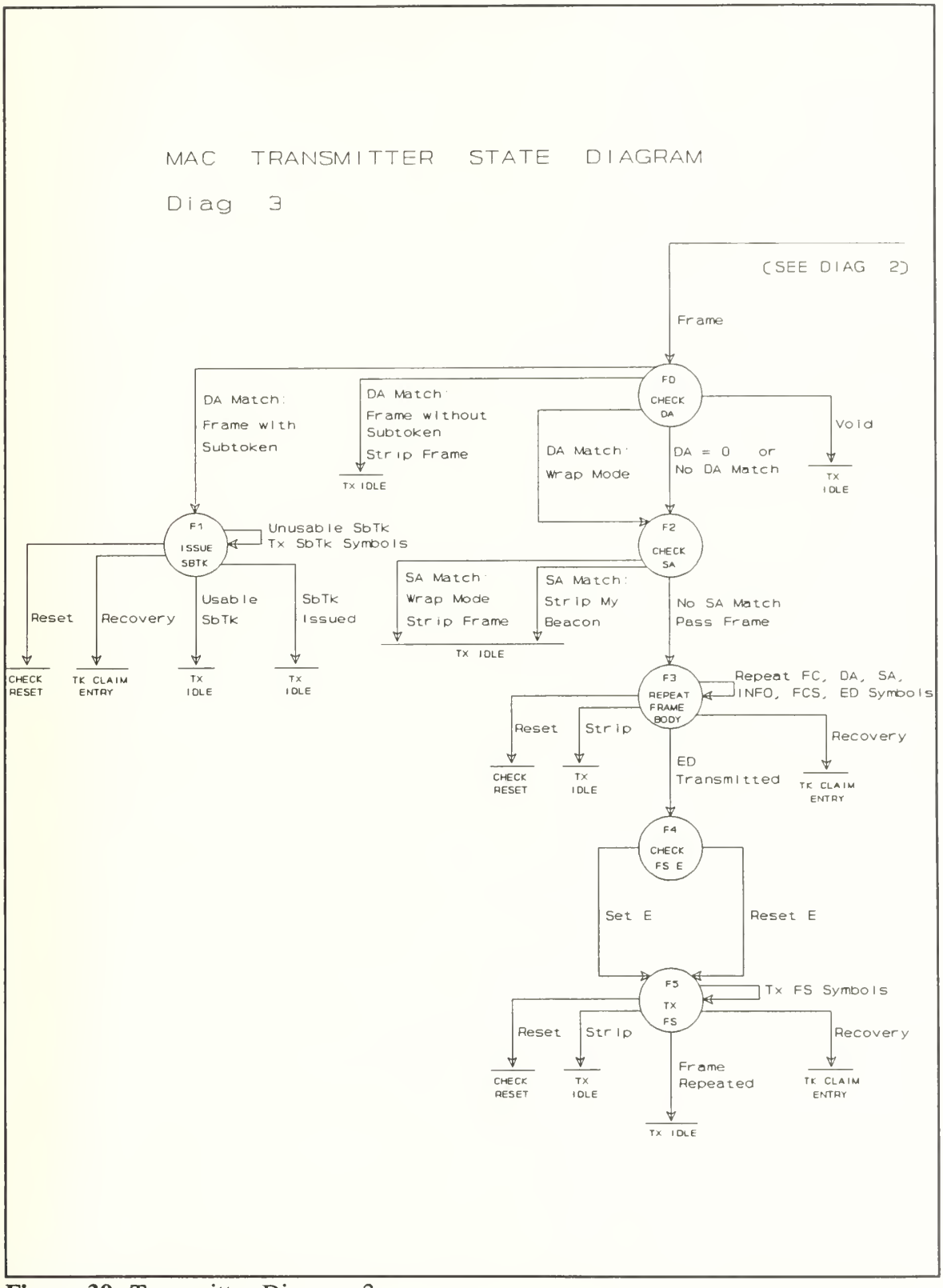


Figure 39: Transmitter Diagram 3

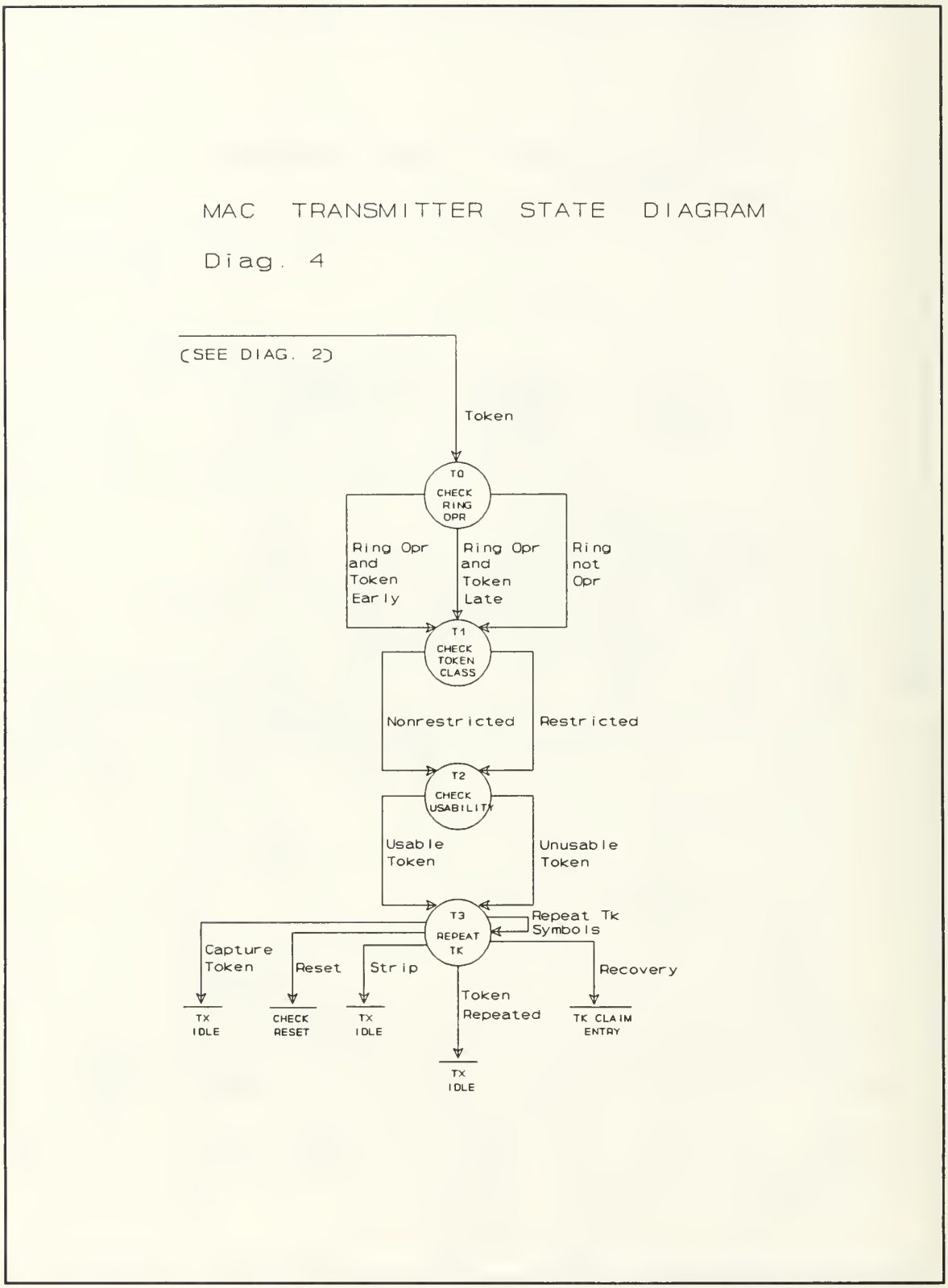


Figure 40: Transmitter Diagram 4

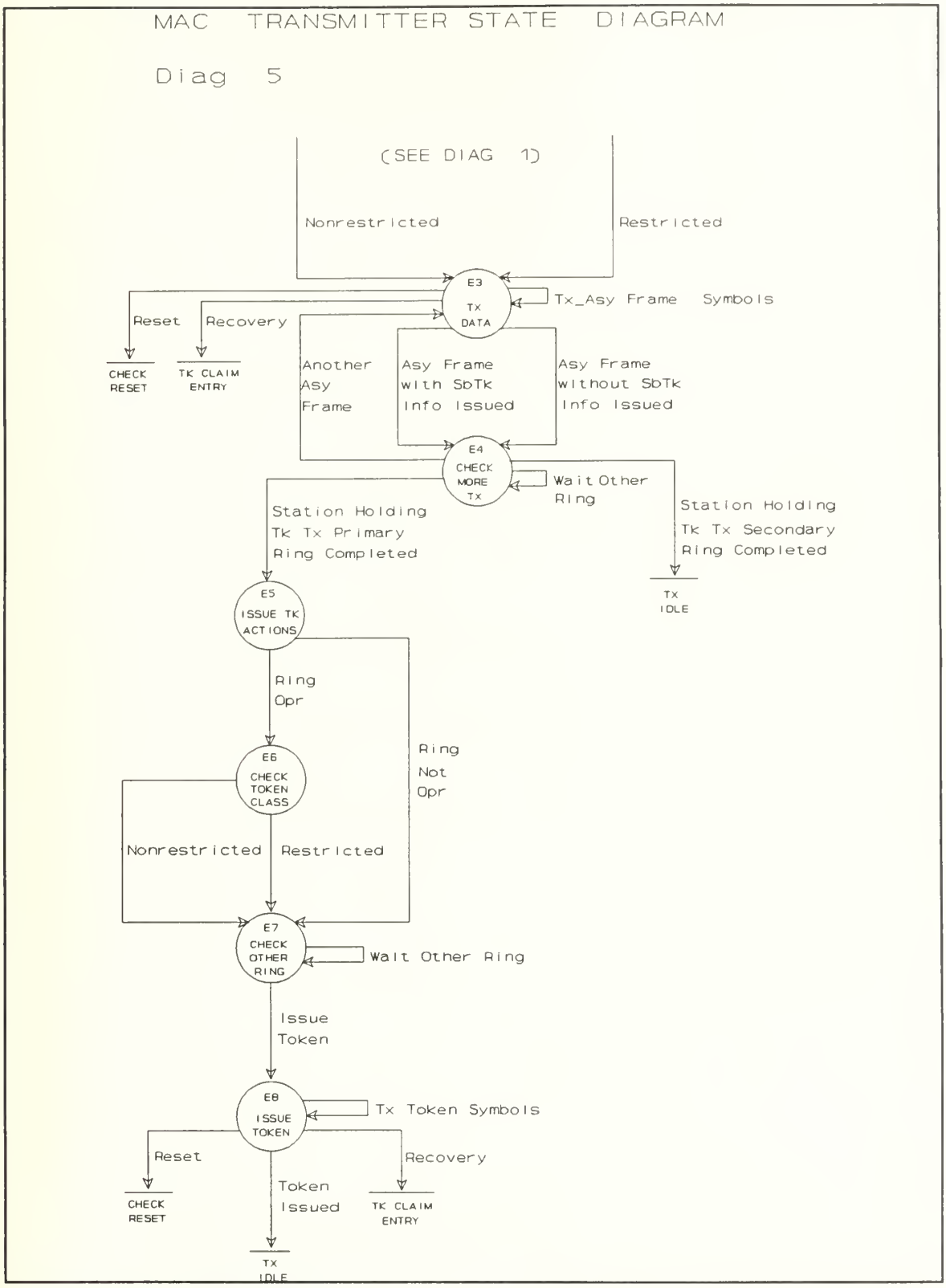


Figure 41: Transmitter Diagram 5

MAC TRANSMITTER STATE DIAGRAM

Diag. 6

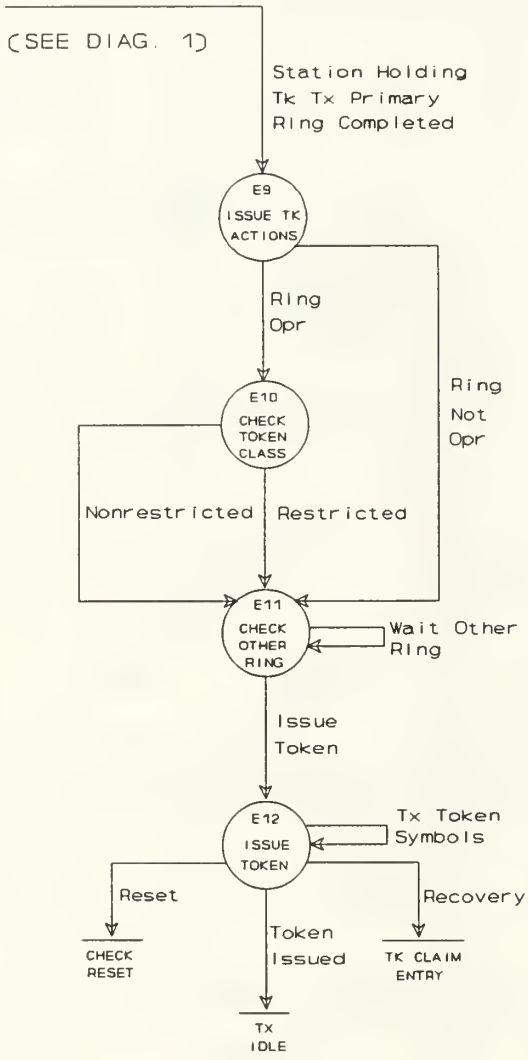


Figure 42:Transmitter Diagram 6

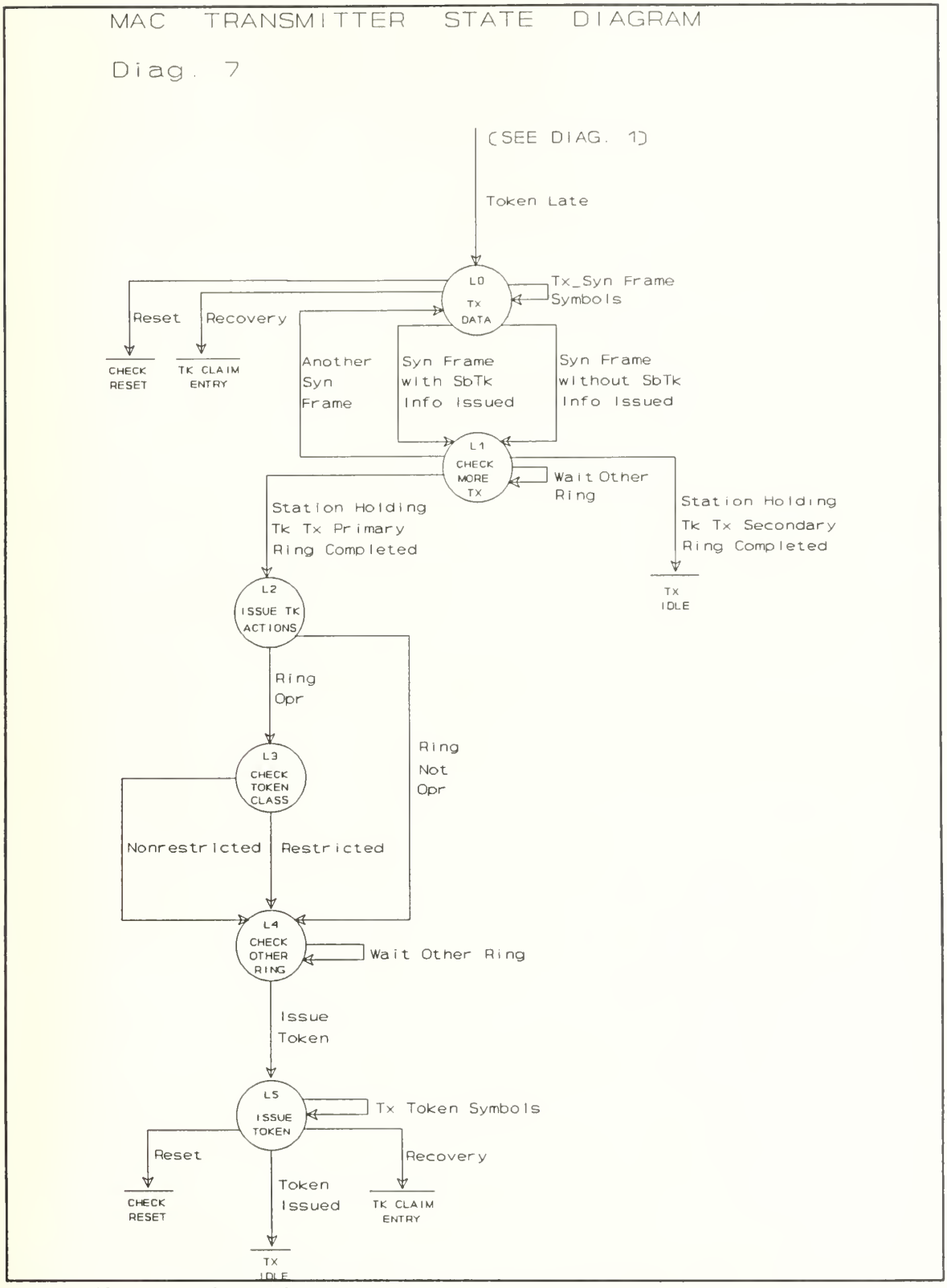


Figure 43: Transmitter Diagram 7

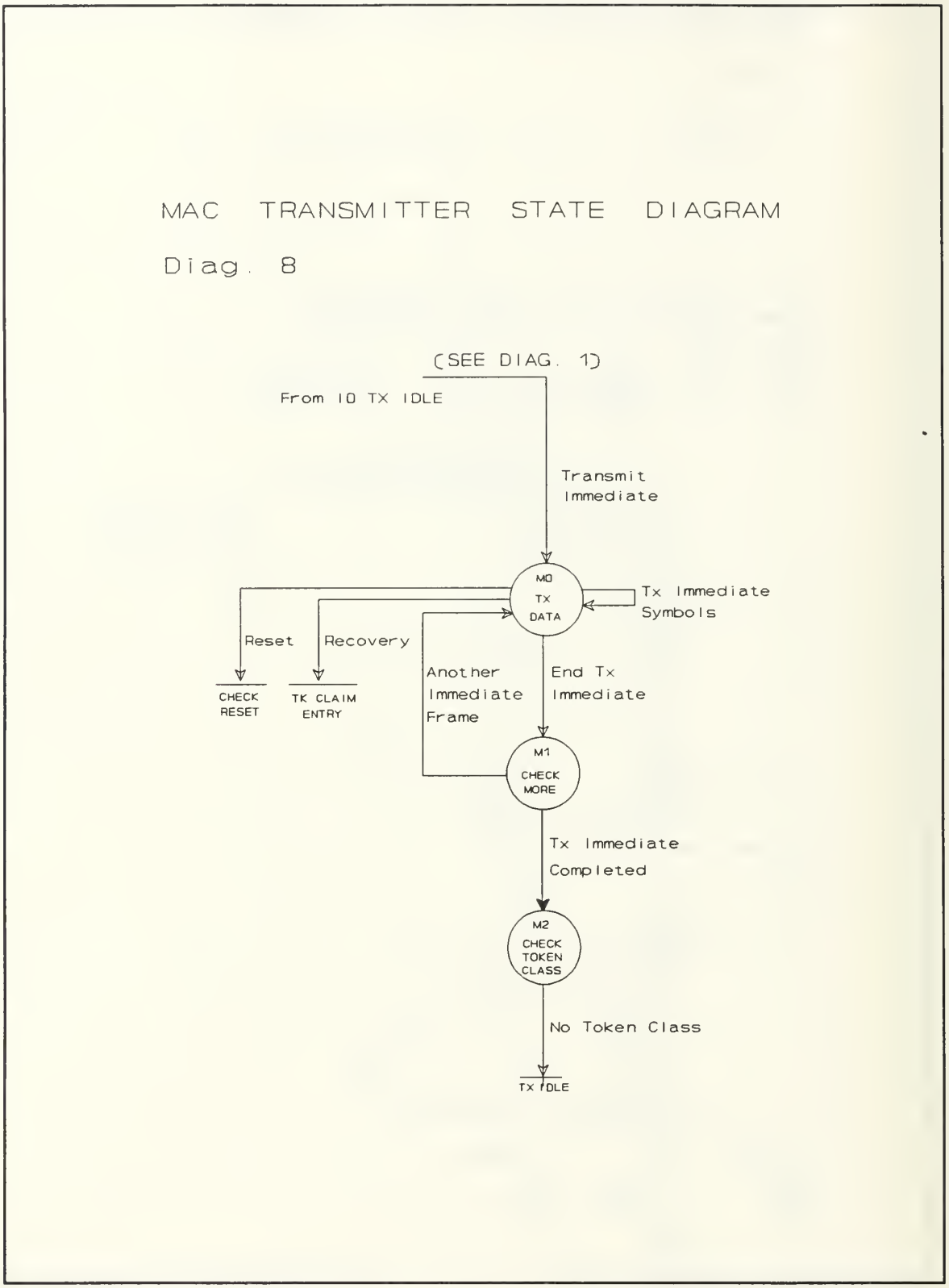


Figure 44: Transmitter Diagram 8

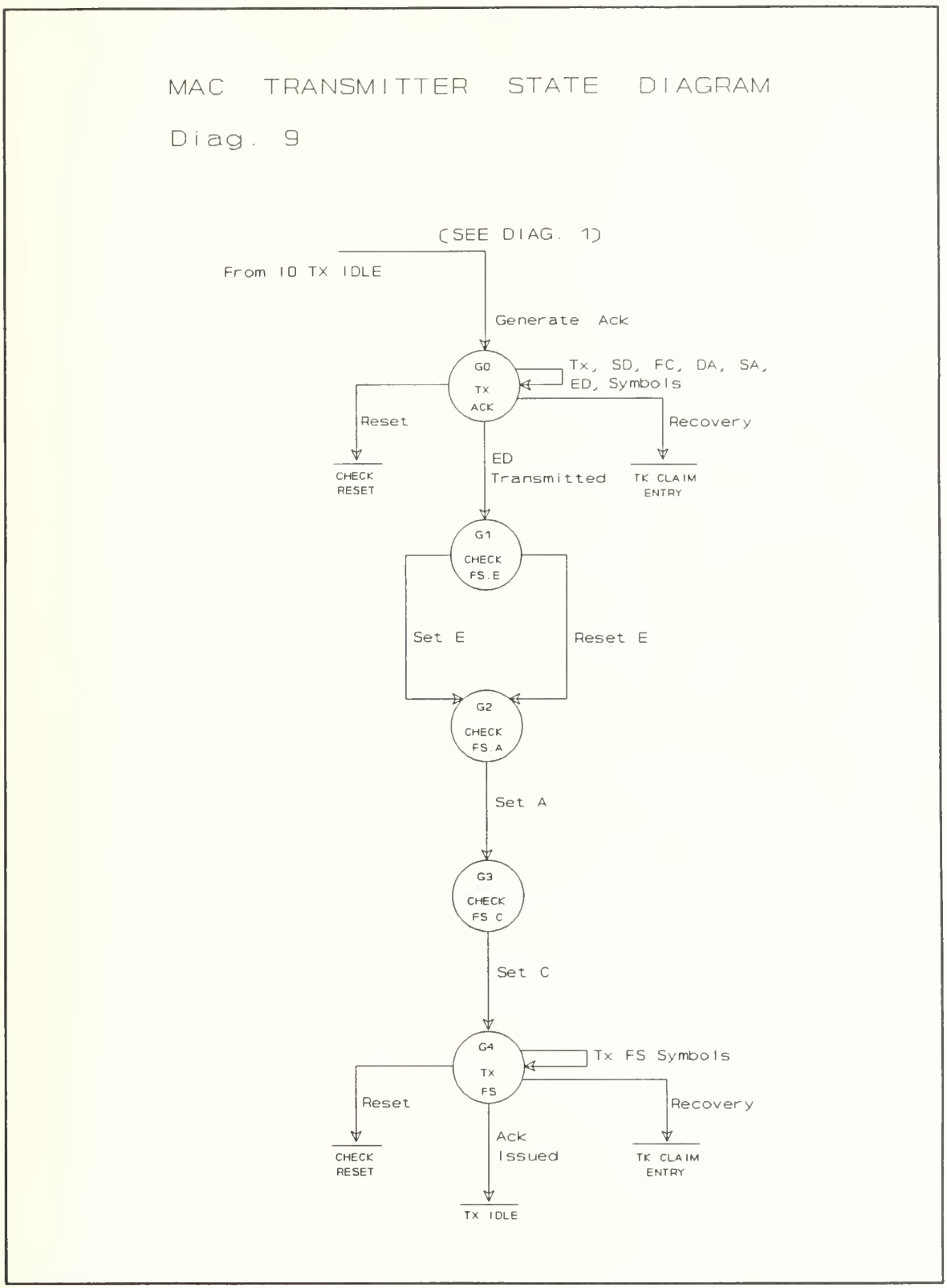


Figure 45: Transmitter Diagram 9

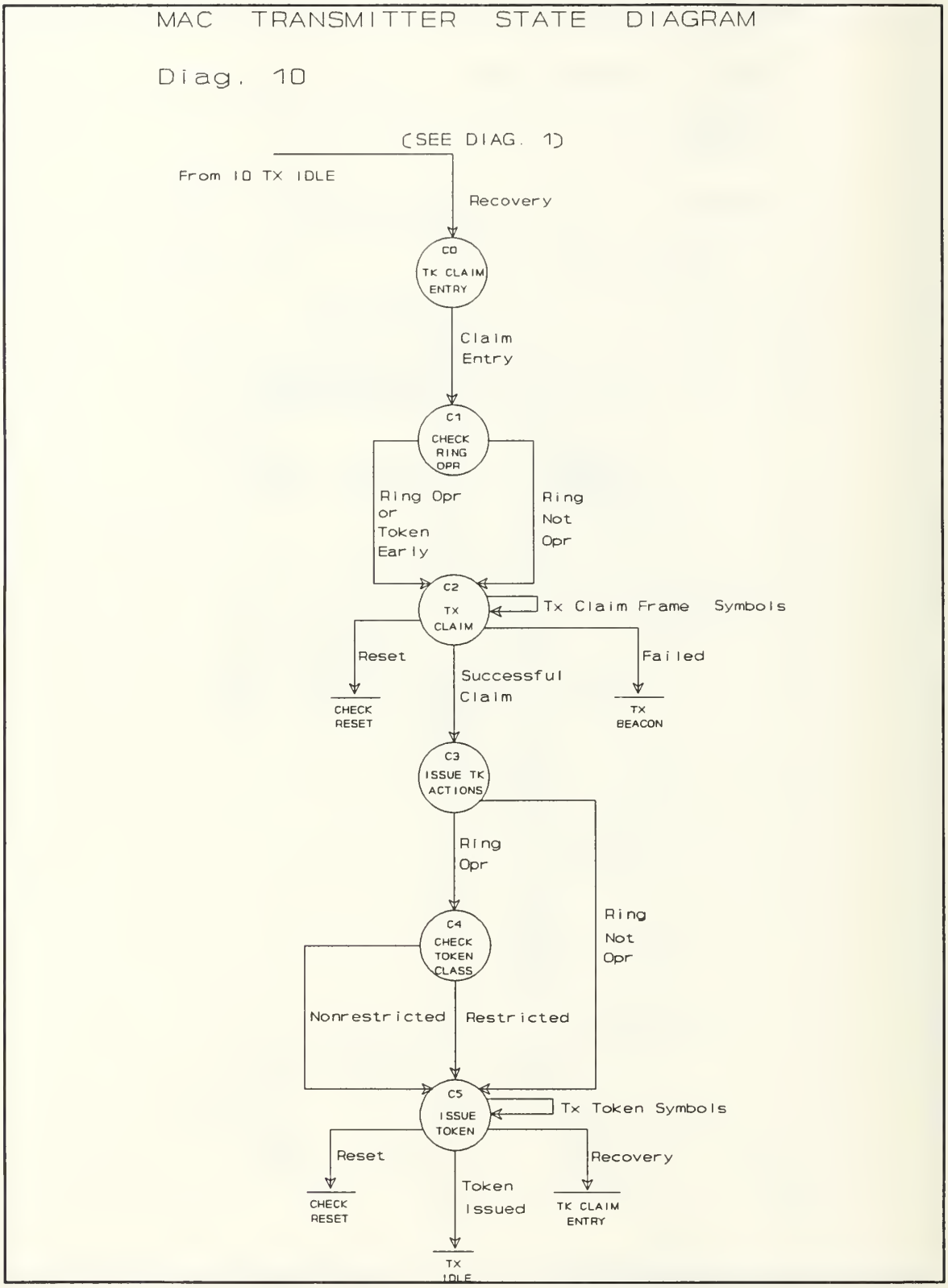


Figure 46: Transmitter Diagram 10

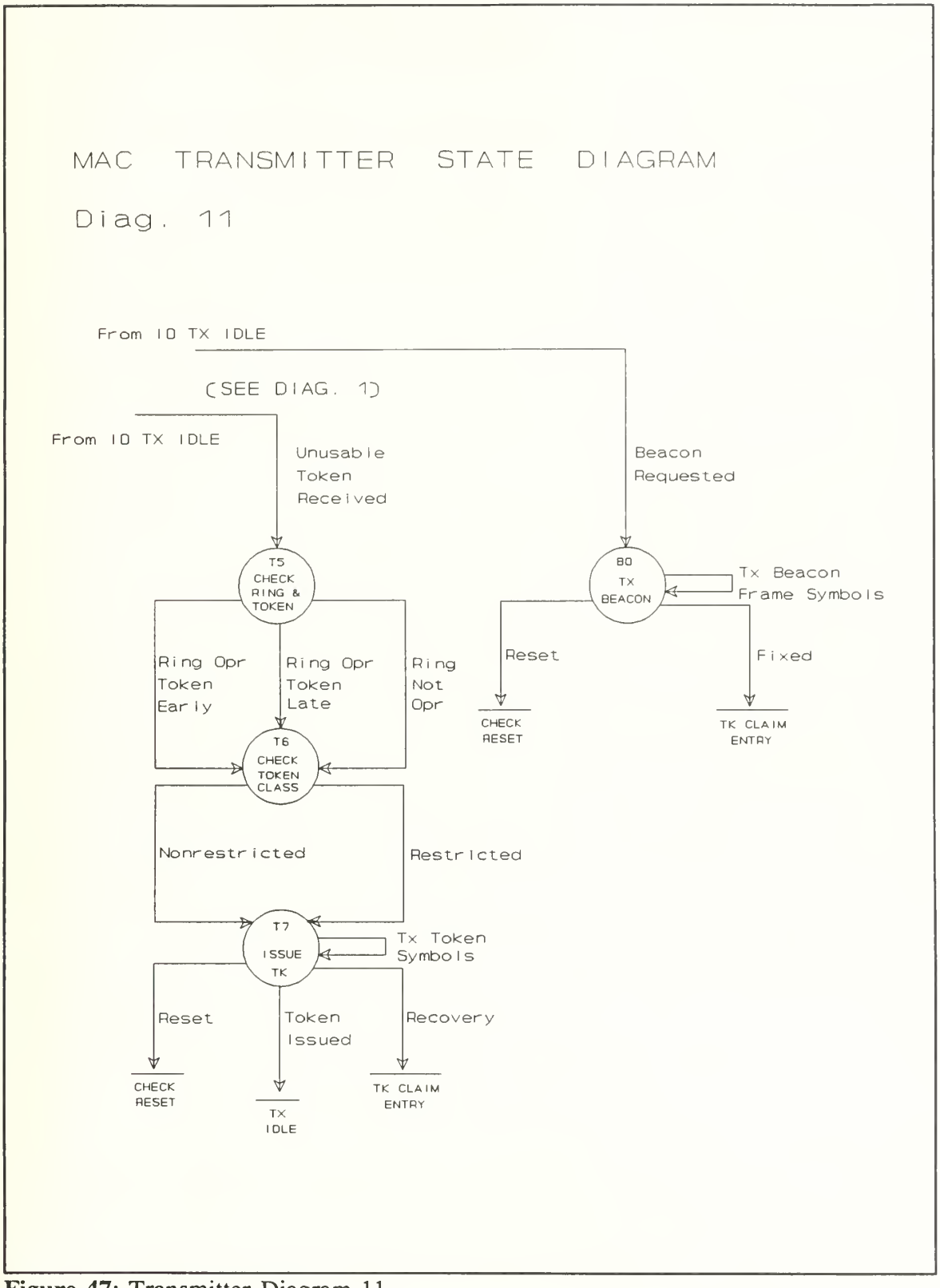


Figure 47: Transmitter Diagram 11

Table 4 is The MAC Transmitter Transition Table. Transmitting symbols transitions show the enabling predicate with the current contents of the PH_request buffer and the current position of the "symbol" pointer on the array of symbols of this buffer.

TABLE 4: MAC TRANSMITTER TRANSITION TABLE

Transition	Enabling Predicate	Action
Tx Idle symbols I(0-0):	Tx_Idle	PH.request[symbol] ← $\{I_1..I_{max}\}$; symbol ← I_1; symbol ← symbol + 1;
Reset x-R0; x=I0,P0,F1,F3, F5,A1,S1,T3,S2 ,E0,E3,E8,E12, L0,L5,L1,L5,M 0,G0,G4,C2,C5 ,T7,B0;	MAC_Reset ∨ SM_MA_CONTROL.request(send_mac_frame) ∨ {MAC_Frame ∧ [Ring_Operational ∨ Late_Ct = 0 ∨ (Token_Class ≠ none ∧ ¬My_Claim)]}	T_Opr ← T_Max; TRT ← T_Opr; Token_Class ← none; Idle ← clear;
Reset_MAC requested R0-I0;	MAC_Reset	Late_Ct ← clear; Ring_Operational ← clear;
Reset MAC frame received R0-I0;	Ring_Operational ∨ Late_Ct = 0	T_Neg ← T_Max; Tx_Idle ← set;
Start I0-P0;	Rc_Start ∧ PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J]$ } ∧ (symbol = $SD_r[J]$)	Idle ← clear; PH_Request(symbol) ← PH_Indication(symbol); symbol ← symbol + 1;
Tx Next SD symbol P(0-0);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$ } ∧ (symbol = $SD_r[K]$)	Tx_Idle ← clear; PH_Request(symbol) ← PH_Indication(symbol); symbol ← symbol + 1;
Strip x-I0; x=P0,F3,F5,A1 ,S1,T3;	Idle ∨ FR_Strip ∨ FO_Error	Tx_Idle ← set;
SFS Transmitted P(0-1);	PH_Request(symbol) = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$ } ∧ (symbol = $SD_x[K]$)	

Transition	Enabling Predicate	Action
Frame P1-F0	PDU_Frame	
Da Match: Frame with Subtoken F(0-1);	(A_Flag) ∧ (FC_r[n,n] = LLC or SMT frame with subtoken) ∧ ¬(Wrap)	
Unusable Sbtk, Tx Sbtk Symbols F(1-1);	Sbtk-buf(symbol) = { $PA_x[I_1..I_{max}]$, SD_x[J,K], FC_x[n,n], SL_x[4n or 12n], EL_x[4n or 12n], $CLASS_x$[2n], $ED_x[T_1,T_2]$ } ∨ ¬(Usable_Sbtk) ∧ (symbol = $PA_x[I_1]$);	PH.request[symbol] ← Subk-buf(symbol); symbol ← symbol + 1;
Recovery x-C0; x=F1,F3,F5,A1,S1,T3,S2,E0,E3,E8,E12,L0,L5,L1,L5,M0,G0,G4,C2,C5,T7	TVX expired ∨ [TRT expires ∧ Late_Ct > 0] ∧ Lower_Claim	
Usable Subtoken F1-I0;	Usable_Sbtk	Tx_Idle ← set;
Sbtk Issued F1-I0;	PH_Request(symbol) = { $PA_x[I_1..I_{max}]$, SD_x[J,K], FC_x[n,n], SL_x[4n or 12n], EL_x[4n or 12n], $CLASS_x$[2n], $ED_x[T_1,T_2]$ } ∧ (symbol = $ED_x[T_2]$)	
Da Match: Frame without Subtoken, Strip Frame F0-I0;	(A_Flag) ∧ (FC_r[n,n] = LLC or SMT frame without subtoken) ∧ ¬(Wrap)	Tx_Idle ← set;
Da Match: Wrap mode F(0-1);	(A_Flag) ∧ (Wrap)	
Da = 0 or No DA Match F(0-2);	(DA_r[4n ∨ 12n] = null) ∧ ¬(A_Flag)	
Void F0-I0;	FC_r[n,n] = Void	Tx_Idle ← set;
SA Match: Strip My MAC Beacon F2-I0;	(DA_r[4n ∨ 12n] = null) ∧ (M_Flag) ∨ (FC_r[n,n] = Beacon Frame)	Tx_Idle ← set;

Transition	Enabling Predicate	Action
No SA Match: Pass Frame F(2-3);	(M_Flag)	
Repeat FC, DA, SA, INFO, FCS, ED Symbols F(3-3);	PH_Indication = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$ } $\wedge$ (symbol = $FC_r[n,n]$)	PH_Request(symbol) ← PH_Indication(symbol); symbol ← symbol + 1;
ED Transmitted F(3-4);	PH_Request = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$, $FC_x[n,n]$, $DA_x[4n \vee 12n]$, $SA_x[4n \vee 12n]$, $INFO_x[n_1..n_{max}]$, $FCS_x[8n]$, $ED_x[T]$ } $\wedge$ (symbol = $ED_x[T]$)	
Set E F(4-5);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$ } $\vee$ (symbol = $FS.E_r[S/R]$) $\wedge$ (E_Flag)	$FS.E_x[S/R]$ ← S;
Reset E F(4-5);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $INFO_r[n_1..n_{max}]$, $FCS_r[8n]$, $ED_r[T]$, $FS.E_r[S/R]$ } $\vee$ (symbol = $FS.E_r[S/R]$) $\wedge$ $\neg$(E_Flag)	$FS.E_x[S/R]$ ← R;
Tx FS Symbols F(5-5);	PH_Request = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$, $FC_x[n,n]$, $DA_x[4n \vee 12n]$, $SA_x[4n \vee 12n]$, $INFO_x[n_1..n_{max}]$, $FCS_x[8n]$, $ED_x[T]$, $FS.E^x[S/R]$, $FS.A_x[S/R]$, $FS.C_x[S/R]$ } $\wedge$ (symbol = $ED_x[T]$)	symbol ← symbol + 1;
Frame Repeated F5-I0;	PH_Request = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$, $FC_x[n,n]$, $DA_x[4n \vee 12n]$, $SA_x[4n \vee 12n]$, $INFO_x[n_1..n_{max}]$, $FCS_x[8n]$, $ED_x[T]$, $FS.E^x[S/R]$, $FS.A_x[S/R]$, $FS.C_x[S/R]$ } $\wedge$ (symbol = $FS.C_x[S/R]$)	Tx_Idle ← set;
Ack P1-A0;	PDU_Ack $\wedge$ $\neg$(Wrap)	
DA Match A0-I0;	A_Flag	Tx_Idle ← set;
No DA Match: Pass Ack A(0-1);	$\neg$(A_Flag)	
Repeat Ack Symbols A(1-1);	PH_Indication(symbol) = { $PA_r[I_1..I_{max}]$, $SD_r[J,K]$, $FC_r[n,n]$, $DA_r[4n \vee 12n]$, $SA_r[4n \vee 12n]$, $ED_r[T]$, $FS.E_r[S/R]$, $FS.A_r[S/R]$, $FS.C_r[S/R]$ } $\wedge$ (symbol = $FC_r[n,n]$)	PH_Request(symbol) ← PH_Indication(symbol); symbol ← symbol + 1;

Transition	Enabling Predicate	Action
Ack Repeated A1-I0;	PH_Request = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$, $FC_x[n,n]$, $DA_x[4n \vee 12n]$, $SA_x[4n \vee 12n]$, $ED_x[T]$, $FS.E^x[S/R]$, $FS.A_x[S/R]$, $FS.C_x[S/R]$ } $\wedge$ (symbol = $FS.C_x[S/R]$)	Tx_Idle ← set;
Subtoken P1-S0;	PDU_Sbtk $\wedge$ ¬(Wrap)	
Usable Subtoken S(0-1);	Ring_Operational $\wedge$ MA_SbTk_Limits $\wedge$ {[Syn_Req_Queued $\vee$ (Asy_Req_Queued $\wedge$ Req_Token_Class = nonrestricted) $\wedge$ $SL_r \leq$ PDU_Q.DA $\leq EL_r$] $\wedge$ [Req_PDU_Queued_Length $\leq CLASS_r$]}	Usable_SbTk ← true;
Unusable Subtoken S(0-1);	¬Ring_Operational $\vee$ ¬MA_SbTk_Limits $\vee$ ¬(Syn_Req_Queued $\vee$ Asy_Req_Queued) $\vee$ {[Syn_Req_Queued $\vee$ (Asy_Req_Queued $\wedge$ Req_Token_Class = nonrestricted) $\wedge$ (PDU_Q.DA $< SL_r \vee$ PDU_Q.DA $> EL_r$)] $\wedge$ [Req_PDU_Queued_Length $> CLASS_r$]}	Usable_SbTk ← false;
Repeat Subtoken Symbols S(1-1);	PH_Indication(symbol) = { $PA_r[I_1..I_n]$, $SD_r[J,K]$, $FC_r[n,n]$, SL_r[4n or 12n], EL_r[4n or 12n], $CLASS_r[n,n]$, $ED_r[T,T]$ } $\wedge$ ¬(Usable_Sbtk) $\wedge$ ¬(EL_r[4n or 12n] = MA[4n or 12n] $\wedge$ (symbol = $FC_r[n,n]$)	PH_Request[symbol] ← PH_Indication(symbol); symbol ← symbol + 1;
Capture Usable Subtoken S1-I0;	Usable_SbTk	Tx_Idle ← set;
Remove Unusable Sbtk S1-I0;	¬Usable_Sbtk $\wedge$ EL_r = MA	Tx_Idle ← set;
Subtoken Repeated S1-I0;	PH_Request = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$, $FC_x[n,n]$, $SL_x[4n \vee 12n]$, $EL_x[4n \vee 12n]$, $CLASS_x[n,n]$, $ED_x[T_1,T_2]$, $\wedge$ (symbol = $ED_x[T_2]$)	Tx_Idle ← set;
Usable Subtoken Received I0-S2;	Usable_SbTk $\wedge$ SbTk_Received	STHT ← enabled; STHT ← $CLASS_r[n,n]$;

Transition	Enabling Predicate	Action
Tx Syn/Asy Frame Symbols S(2-2);	(S-buf[symbol] ∨ A-buf[symbol]) = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = $SD_x[J]$)	Tx_Idle ← clear; PH_Request[symbol] ← (S-buf[symbol] ∨ A-buf[symbol]); symbol ← symbol + 1;
T(29-30): End Syn or Asy Frame S(2-3);	PH_Request(symbol) = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = $FS.C_x$[S/R])	
Another Frame S(3-2);	((S-buf[symbol] ∨ A-buf[symbol]) ≠ null) ∧ Usable_SbTk	
Station Holding SbTk Tx Completed S3-I0;	STHT expires ∨ [(Syn_Req_Queued > Remaining_Time) ∧ (Asy_Req_Queued > Remaining_Time)] ∨ (¬Syn_Req_Queued ∧ ¬Asy_Req_Queued)	Tx_Idle ← set;
Token P1-T0;	PDU_Tk	
Ring Opr and Token Early T(0-1); T(5-6);	Ring_Operational ∧ Late_Ct = 0	TRT ← T_Opr;
Ring Opr and Token Late T(0-1); T(5-6);	Ring_Operational ∧ Late_Ct > 0	Late_Ct ← 0;
Ring not Opr T(0-1); T(5-6);	¬(Ring_Operational)	T_Opr ← T_Neg; TRT ← T_Opr; Late_Ct ← 1; Ring_Operational ← set;
Nonrestricted T(1-2); T(6-7);	$FC_r.L = 0$	Token_Class ← nonrestricted
Restricted T(1-2); T(6-7);	$FC_r.L = 1$	Token_Class ← restricted

Transition	Enabling Predicate	Action
Usable Token T(2-3);	Ring_Operational ∧ {[Syn_Req_Queued ∧ Syn_Allowed] ∨ [Late_Ct = 0 ∧ (Asy_Req_Queued ∧ Req_Token_Class = FC_r.L ∧ (Non_priority_Request ∨ TRT < T_Pri(Request_Priority) ∧ (Nonrestricted Request ∧ (B_Flag ∧ Req_Token_Class = restricted))]}	Usable_Tk ← true;
Unusable Token T(2-3);	¬Ring_Operational ∨ ¬{[Syn_Req_Queued ∧ Syn_Allowed] ∨ [Late_Ct = 0 ∧ (Asy_Req_Queued ∧ Req_Token_Class = FC_r.L ∧ (Non_priority_Request ∨ TRT < T_Pri(Request_Priority) ∧ (Nonrestricted Request ∧ (B_Flag ∧ Req_Token_Class = restricted))]}	Usable_Tk ← false;
Repeat Token Symbols T(3-3);	PH_Indication(symbol) = { $PA_r[I_1..I_n]$, $SD_r[J,K]$, $FC_r[n,n]$, SL_r[4n or 12n], $ED_r[T_1,T_2]$ } ∧ ¬(Usable_Tk) ∧ (symbol = $FC_r[n,n]$)	PH_Request[symbol] ← PH_Indication(symbol); symbol ← symbol + 1;
Capture Token T3-I0;	Usable_Tk	Tx_Idle ← set;
Token Repeated T3-I0;	PH_Request = { $PA_x[I_1..I_{max}]$, $SD_x[J,K]$, $FC_x[n,n]$, $ED_x[T_1,T_2]$, ∧ (symbol = $ED_x[T_2]$)	Tx_Idle ← set;
Usable Token Received I0-T4;	Usable_Tk ∧ Tk_Received	THT ← disabled;
Token Early T4-E0;	Late_Ct = 0	THT ← TRT; TRT ← T_Opr;
Token Late T4-L0;	Late_Ct > 0	THT ← expired; Late_Ct ← 0;
Tx Syn Frame symbols E(0-0); L(0-0);	S-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], $ED_x[T]$, $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = $SD_x[J]$)	Tx_Idle ← clear; PH_Request[symbol] ← S- buf[symbol]; symbol ← symbol + 1;

Transition	Enabling Predicate	Action
Syn Frame with Sbtk Info Issued E(0-1); L(0-1);	PH_request = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ ($FC_x[n,n]$ = LLC or SMT Syn Frame with Subtoken) ∧ (symbol = $FS.C_x$[S/R])	
Syn Frame without Sbtk Info Issued E(0-1); L(0-1);	PH_request = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ ($FC_x[n,n]$ = LLC or SMT Syn Frame without Subtoken) ∧ (symbol = $FS.C_x$[S/R])	
Another Syn Frame E(1-0); L(1-0);	(S-buf[symbol]) ≠ null) ∧ Usable_Tk	
Station Holding Tk Tx Secondary Ring Completed E1-I0;E4-I0; L1-I0;	TRT expires ∨ (synchronous allowed ∧ Syn_Req_Queued > Syn_band_allocation) ∨ ¬Usable_Tk	$Wait_{secondary}$ ← false;
Wait Other Ring E(1-1); E(4-4); L(1-1); L(4-4); E(7-7); E(11-11);	$Wait_{primary}$ ∧ $Wait_{secondary}$	Tx_Idle ← set;
Asy Service E1-E2;	¬Usable_Tk ∧ ¬Syn_Req_Queued	
Nonrestricted E2-E3;	Token_Class = nonrestricted	
Restricted E2-E3;	Token_Class = restricted	
Tx Asy Frame symbols E(3-3);	A-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = SD_x[J])	PH_Request[symbol] ← A-buf[symbol]; symbol ← symbol + 1;

Transition	Enabling Predicate	Action
Asy Frame with Sbtk Info Issued E(3-4);	PH_request = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, $DA_x[4n$ or $12n]$, $SA_x[4n$ or $12n]$, $INFO_x[n_1..n_{max}]$, $FCS_x[8n]$, $ED_x[T]$, $FS.E_x[R/S]$, $FS.A_x[S/R]$, $FS.C_x[S/R]$ } ∧ ($FC_x[n,n]$ = LLC or SMT Asy Frame with Subtoken) ∧ (symbol = $FS.C_x[S/R]$)	
Syn Frame without Sbtk Info Issued E(3-4);	PH_request = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, $DA_x[4n$ or $12n]$, $SA_x[4n$ or $12n]$, $INFO_x[n_1..n_{max}]$, $FCS_x[8n]$, $ED_x[T]$, $FS.E_x[R/S]$, $FS.A_x[S/R]$, $FS.C_x[S/R]$ } ∧ ($FC_x[n,n]$ = LLC or SMT Asy Frame without Subtoken) ∧ (symbol = $FS.C_x[S/R]$)	
Another Asy Frame E(4-3);	(A-buf[symbol]) ≠ null) ∧ Usable_Tk ∧	
Station Holding Tk Tx Primary Ring Completed E(1-9); E(4-5);	TRT expires ∨ (synchronous allowed ∧ Syn_Req_Queued > Syn_band_allocation) ∨ ¬Usable_Tk	$Wait_{primary}$ ← false;
Ring Opr E(5-6); E(9-10); L(2-3); C(3-4);	Ring_operational	
Nonrestricted E(6-7); E(10-11); L(3-4); C(4-5);	(Token_Class = nonrestricted) ∨ ¬(R_Flag)	B_Flag ← set;
Restricted E(6-7); E(10-11); L(3-4); C(4-5);	(Token_Class = restricted) ∨ R_Flag	B_Flag ← clear;
Ring Not Opr E(5-7); E(9-11); L(3-5); C(3-5);	¬(Ring_Operational)	T_Opr ← T_Neg; TRT ← T_Opr; Late_Ct ← 1; Ring_Operational ← set;
Issue Token E(7-8);	($Wait_{primary}$ = false) ∧ ($Wait_{secondary}$ = false)	

Transition	Enabling Predicate	Action
Tx Token Symbols E(8-8); E(12-12); L(5-5); C(5-5); T(7-7);	Tk-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, $ED[T_1,T_2]$ } ∧ (symbol = $PA_x[I_1]$) ∧ ¬Usable_Tk	PH_Request[symbol] ← Tk-buf[symbol]; symbol ← symbol + 1;
Token Issued x-I0; x=E8,E12,L5, C5,T7;	PH_Request[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, $ED[T_1,T_2]$ } ∧ (symbol = $ED[T_2]$)	Tx_Idle ← set;
Transmit Immediate I0-M0;	requested_service_class = Immediate ∧ ¬Ring_Operational ∧ Token_Class = none	TRT ← T_Opr; Late_Ct ← 0; Tx_Idle ← clear;
Tx Immediate Symbols M(0-0);	I-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = $SD_x[J]$)	Tx_Idle ← clear; PH_Request[symbol] ← I-buf[symbol]; symbol ← symbol + 1;
End Tx Immediate M1-M2;	I-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], ED_x[T], $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = $FS.C_x$[S/R])	PH_Request[symbol] ← I-buf[symbol];
Another Immediate Frame M(1-0);	I-buf[symbol] ≠ null	
Tx Immediate Completed M1-M2;	I-buf[symbol] = null	
No Token Class M2-I0;	Token_Class = none	Tx_Idle ← set; TRT ← T_Opr; Late_Ct ← 1;
Generate Ack I0-G0;	Ack_Frame	
Tx SD,FC,DA, SA,ED Symbols G0-G0;	Ack-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], ED_x[T] } ∧ (symbol = $SD_x[J]$)	Tx_Idle ← clear; PH_Request[symbol] ← Ack-buf[symbol]; symbol ← symbol + 1;

Transition	Enabling Predicate	Action
ED Transmitted M(1-2);	PH_Request(symbol) = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $ED_x[T]$ } ∧ (symbol = $ED_x[T]$)	
Set E G(1-2);	$PH\_Indication(symbol)_{other\_ring}$ = { $FS.E_r[S/R]$ } ∧ E_Flag	$FS.E_x[S/R] \leftarrow S$;
Reset E G(1-2);	$PH\_Indication(symbol)_{other\_ring}$ = { $FS.E_r[S/R]$ } ∧ ¬E_Flag	$FS.E_x[S/R] \leftarrow R$;
Set A G(2-3);	$PH\_Indication(symbol)_{other\_ring}$ = { $FS.A_r[S]$ } ∨ A_Flag	$FS.A_x[S/R] \leftarrow S$;
Set C G(3-4);	$PH\_Indication(symbol)_{other\_ring}$ = { $FS.C_r[S/R]$ } ∧ C_Flag ∧ ¬N_Flag	$FS.C_x[S/R] \leftarrow S$;
Tx FS symbols G4-G4;	PH_Request(symbol) = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $ED_x[T]$ } ∧ (symbol = $ED_x[T]$)	PH_Request [symbol] ← { $FS.E_x[R/S]$, $FS.A_x[S]$, $FS.C_x[S]$ }
Ack Issued G4-I0;	PH_Request(symbol) = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $ED_x[T]$, $FS.E_x[R/S]$, $FS.A_x[S]$, $FS.C_x[S]$ } ∧ (symbol = $FS.C_x[S/R]$)	Tx_Idle ← set;
Recovery I0-C0;		
Claim Entry C(0-1);	SM_MA_CONTROL.request[Claim]	T_Opr ← T_Max; TRT ← T_Opr; Token_Clas = none;
Tx Claim Frame Symbols C(2-2);	Claim-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], $ED_x[T]$, $FS.E_x[R/S]$, $FS.A_x[S/R]$, $FS.C_x[S/R]$ } ∧ (symbol = $SD_x[J]$)	Tx_Idle ← clear; PH_Request[symbol] ← Claim-buf[symbol]; symbol ← symbol + 1;
Successful Claim C(2-3);	My_Claim	TRT ← T_Opr; Token_Class ← nonrestricted;
Failed C2-B0;	PH_Request[symbol] = Claim-buf[symbol] ∧ TRT expires	Beacon_type ← Unsuccessful Claim; Beacon.DA ← null;
Unusable Token Received I0-T5;	Tk_Received ∧ ¬Usable_Tk	THT ← disabled; Tx_Idle ← clear;

Transition	Enabling Predicate	Action
Beacon Requested I0-B0;	SM_MA.CONTROL.request[Beacon]	Tx_Idle ← clear;
Tx Beacon Frame Symbols B(0-0);	Beacon-buf[symbol] = { $PA_x[I_1..I_n]$, $SD_x[J,K]$, $FC_x[n,n]$, DA_x[4n or 12n], SA_x[4n or 12n], $INFO_x[n_1..n_{max}]$, FCS_x[8n], $ED_x[T]$, $FS.E_x$[R/S], $FS.A_x$[S/R], $FS.C_x$[S/R] } ∧ (symbol = $SD_x[J]$)	Tx_Idle ← clear; PH_Request[symbol] ← Beacon-buf[symbol]; symbol ← symbol + 1;
Fixed B0-I0;	PH_Indication(symbol) ← { Beacon symbols } ∧ My_Beacon	Tx_Idle ← set;

6. Local and Timer Data Type Specification

Table 3 presents the local and timer data type specification. This table contains each data type used in the machines transition tables. Its range, initial value, and purpose are included. Values on/off, true/false, set/reset of boolean variables have the same meaning and can be used interchangeably. Data types marked with (*) are not specified in the FDDI MAC standard.

TABLE 5: LOCAL AND TIMER DATA TYPE SPECIFICATION

Name	Range	Initial value	Purpose
Error_Ct	[0, ∞)	0	Error Counter - Count of reportable frame errors.
Frame_Ct	[0, ∞)	0	Frame Counter - Count of all frames received.
Late_Ct	[0, ∞)	0	Late Counter - Count of TRT expirations (Token Lateness).
Lost_Ct	[0, ∞)	0	Lost Counter - Count of PDU detected as lost.
symbol_ct (*)	[0, ∞)	0	Lost Counter - Count of symbols in a transfer of data.
Ack_ct (*)	[0, ∞)	0	Acknowledgment Counter - Count of acks received.
H_Flag	boolean [set, reset]	reset	Address recognized indicator Flag - Indicates Destinations Address (DA) match in last received frame.
C_Flag	boolean [set, reset]	reset	Frame copied indicator Flag - Indicates successful copying of last received frame.
E_Flag	boolean [set, reset]	reset	Error detected indicator Flag - Indicates error detected in last received frame.
H_Flag	boolean [set, reset]	reset	Higher address Flag - Indicates Higher Source Address (HSA) received.
L_Flag	boolean [set, reset]	reset	Lower Address Flag - Indicates Lower Source Address (LSA) received.
M_Flag	boolean [set, reset]	reset	My address Flag - Indicates My Source Address (MSA) received.

Name	Range	Initial value	Purpose
N_Flag	boolean [set, reset]	reset	Next address Flag - Indicates Next Station (NSA) Addressing.
R_Flag	boolean [set, reset]	reset	Restricted class Flag - Indicates restricted Token_class for the last valid Token_received.
Rc_Start	boolean [true, false]	false	Received start signal - Indicates the starting of an incoming PDU.
T_Req	integer number of symbol times at 100 Mbits/sec [T_Min, T_Max]	0	Requested TTRT - Indicates the Requested TTRT for this MAC's synchronous traffic to negotiate for the lowest value of T_Opr.
T_Bid_Rc	integer number of symbol times at 100 Mbits/sec [T_Min, T_Max]	0	Bidding TTRT Received - Indicates the Bidding TTRT received by this station in Claim Frames.
T_Max	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Maximum TTRT - Indicates the Maximum TTRT to be supported by this station.
T_Min	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Minimum TTRT - Indicates the Minimum TTRT to be supported by this station.
T_Neg	integer number of symbol times at 100 Mbits/sec [T_Min, T_Max]	T_Opr = lowest value of T_Req	Negotiated TTRT - Indicates the Negotiated TTRT during Claim process (In receiver). The lowest value of T_Req becomes T_Opr for the ring.
T_Opr	integer number of symbol times at 100 Mbits/sec [T_Min, T_Max]	T_Max	Operative Value of TTRT - Indicates the Operative value of TTRT for this station (in transmitter).
T_Pri(Request_p)	integer [0, ∞)	0	Priority Request Values - Set of Request priority Token rotation time thresholds.
Token_Class	3 values	0	Class of token Restricted, nonrestricted, and none
THT	integer number of symbol times at 100 Mbits/sec [T_Min, T_Max]	current value of TRT	Token-Holding Timer - Controls the time of a MAC Asynchronous frame transmission.

Name	Range	Initial value	Purpose
TRT	integer number of symbol times at 100 Mbits/sec [T_min, T_Max]	current value of T_Opr	Token-Rotation Timer - Controls ring scheduling during normal operation and is used to detect and recover from serious ring error situations.
TTRT	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Target-Token Rotation Time - Indicates the various times that can be assumed during different MAC processes (T_Req, T_Bid_Rc, T_Max, T_Min, T_Neg, T_Opr).
TVX	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Valid-Transmission Timer - To recover from transient ring error situations.
STHT(*)	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Subtoken Holding Timer - Controls the time of the subtoken data transmission to ensure controlled concurrent access.
Idle(*)	boolean [true, false]	false	Idle signal - The Receiver generates this signal whenever the incoming symbol is an idle symbol.
Tx_Idle(*)	boolean [true, false]	false	Tx Idle variable - The Transmitter sets this local variable whenever it is requested to enter the TX IDLE state. This occurs due to an Idle, or Fr_Strip, or FO_error signal from the receiver or after completion of a PDU transmission.
FO_Error	boolean [true, false]	false	Format Error signal - The Receiver generates this signal whenever an incoming symbol does not conform with a sequence specified in the FDDI MAC standard.
FR_Strip	boolean [true, false]	false	Frame Strip signal - The Receiver generates this signal whenever an incoming symbol sequence indicates a condition for removing a PDU from the ring.
PDU_Frame(*)	boolean [true, false]	false	Frame signal - The Receiver generates this signal when it scans the FC field of the incoming PDU and recognizes that this PDU has a format of Frame.

Name	Range	Initial value	Purpose
PDU_Ack(*)	boolean [true, false]	false	Ack signal - The Receiver generates this signal when it scans the FC field of the incoming PDU and recognizes that this PDU has a format of Acknowledgment.
PDU_Tk(*)	boolean [true, false]	false	Token signal - The Receiver generates this signal when it scans the FC field of the incoming PDU and recognizes that this PDU has a format of Token.
PDU_Sbtk(*)	boolean [true, false]	false	Subtoken signal - The Receiver generates this signal when it scans the FC field of the incoming PDU and recognizes that this PDU has a format of Subtoken.
Copy_Frame(*)	boolean [true, false]	false	Copy Frame variable - The Receiver set this variable when a frame is to be copied for the local entity (e.g., an LLC or frame addressed to this station).
FR_Received	boolean [true, false]	false	Frame Received signal - The Receiver generates this signal when a PDU of Frame format is received.
Ack_Frame(*)	boolean [true, false]	false	Ack Frame signal - The Receiver generates this signal for the transmitter on the opposite ring when a PDU of Frame format was received and copied locally into the receive buffer.
Copy_Ack(*)	boolean [true, false]	false	Copy acknowlegment - The Receiver set this variable when an acknowledgment is to be copied to save the frame status report for the local entity.
My_Claim	boolean [true, false]	false	My Claim signal - The Receiver generates this signal to indicate an incoming claim frame with this MAC's own bid.
Higher_Claim	boolean [true, false]	false	Higher Claim signal - The Receiver generates this signal to indicate an incoming claim frame with a MAC higher bid.

Name	Range	Initial value	Purpose
Lower_Claim	boolean [true, false]	false	Lower Claim signal - The Receiver generates this signal to indicate an incoming claim frame with a MAC lower bid.
My_Beacon	boolean [true, false]	false	My Beacon signal - The Receiver generates this signal to indicate an incoming MAC's own beacon frame.
Other_Beacon	boolean [true, false]	false	Other Beacon signal - The Receiver generates this signal to indicate an incoming beacon frame from other MAC.
Other_LLC_or_SMT (*)	boolean [true, false]	false	Other LLC or SMT frame signal - The Receiver generates this signal to indicate that an incoming frame is an LLC or SMT frame addressed to other station.
Sbtk_Flag(*)	boolean [set, reset]	set	Subtoken Flag - Indicates if an incoming subtoken is usable or not.
Sbtk_Received(*)	boolean [true, false]	false	Subtoken Received signal - The Receiver generates this signal to indicate that a subtoken has been received.
Usable_tk(*)	boolean [true, false]	false	Usable Token - The Transmitter uses this variable to check if the token can be captured.
Usable_Sbtk(*)	boolean [true, false]	false	Usable Subtoken - The Transmitter uses this variable to check if the subtoken can be captured.
Tk_Received	boolean [true, false]	false	Token Received signal - The Receiver generates this signal to indicate that a token has been received.
Ring_Operational	boolean [true, false]	true	Ring Operational - Indicates the operational status of the ring.
Wrap(*)	boolean [true, false]	true	Wrap configuration variable Indicates that a serious physical failure has occurred and the configuration changed to one logical ring. This is an overall network-controlling function.
$Wait_{primary}$(*)	boolean [true, false]	true	Wait primary ring transmission- During simultaneous transmission indicates that primary ring is transmitting.
$Wait_{secondary}$(*)	boolean [true, false]	true	Wait secondary ring transmission- During simultaneous transmission indicates that secondary ring is transmitting.

Name	Range	Initial value	Purpose
Syn_Req_Queued	boolean [true, false]	false	Synchronous Request Queued - Indicates a request for transmission of synchronous PDU queued.
Syn_band_allocation	boolean [true, false]	false	Synchronous Bandwidth Allocation - Indicates the alloted bandwidth for the stations during the synchronous service.
Asy_Req_Queued	boolean [true, false]	false	Synchronous Request Queued - Indicates a request for transmission of asynchronous PDU queued.
Remaining_Time(*)	integer number of symbol times at 100 Mbits/sec [0, max)	0	Remaining time of subtoken duration- Indicates the amount of time available after transmission of frames.
Req_Q(length)(*)	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Requested Queued Length - Indicates the length in number of symbol times (duration) of the requested PDU queued for transmission.
Sbtk_Class(*)	integer number of symbol times at 100 Mbits/sec [0, ∞)	0	Subtoken Class - Indicates the length in number of symbol times (duration) of the Subtoken. Class is a stepwise function of the length.
PH_Invalid	boolean [true, false]	false	PH_Invalid is the signal parameter of the PHY to MAC Invalid Indication primitive - When this occurs the PHY entity is unable to present a valid symbol to MAC and the Receiver enters into its LISTEN state.
SM_MA_ CONTROL. request(beacon)	boolean [true, false]	false	SMT to MAC Control Request primitive with Beacon signal parameter - When this occurs the Transmitter enters into its TRANSMIT BEACON state.
SM_MA_ CONTROL. request(reset)	boolean [true, false]	false	SMT to MAC Control Request primitive with Reset signal parameter - When this occurs MAC generates the MAC_Reset signal.
Mac_Reset	boolean [true, false]	false	MAC Reset signal - MAC generates this signal in response to a control_action requested by SMT to reset the MAC state machines (Receiver and Transmitter). When it occurs the Receiver enters into its LISTEN state and the transmitter enters into its IDLE state.

Name	Range	Initial value	Purpose
PH_Request (symbol)	array [1..max] of buffer	-	PH_Request Buffer - It is the output buffer at the Transmitter side. It holds each outgoing symbol on a transfer of data from the MAC entity to the PHY entity. This transfer of data occurs whenever a PH_Request signal is sent to PHY whenever MAC has a symbol to output.
PH_Indication (symbol)	array [1..max] of buffer	-	PH_Indication Buffer - input buffer at the Receiver side. It holds each incoming symbol on a transfer of data from the PHY entity to MAC entity. This transfer of data occurs whenever a PH_Indication signal comes from PHY after a symbol is decoded.
S-buf(symbol)(*)	array [1..max] of buffer	-	Synchonous Buffer - buffer for transmission of synchronous queued SDUs.
A-buf(symbol)(*)	array [1..max] of buffer	-	Asynchonous Buffer - buffer for transmission of asynchronous queued SDUs.
Rcv-buf(symbol)(*)	array [1..max] of buffer	-	Receive Buffer - It is the buffer for copying frames to the local entities.
Symbol	[0..MFL]	0	*Symbol* is a pointer to the current symbol on the buffer array. Symbols are: J, K, T, R, S, I, or n and they represent the smallest signalling element used by the DDL entities. For both Receiver and Transmitter machines *symbol* is a generical representation to denote a pointer to the current symbol.
PA[I_1..I_{max}]	field of [I_1..I_{max}] Idle symbols	I_1	Preamble of a PDU format - Contains a variable number of Idle symbols. The subscripts $_r$ and $_x$ stand for received and transmitted respectively.
SD[J,K]	field of J and K symbols		Starting Delimiter of a PDU format - Contains the symbol J followed by K.
DA[4n ∨ 12n]	field of 4n or 12n symbols		Destination Address of a frame format Contains a fixed number of either four or twelve data quartet symbols.
SA[4n ∨ 12n]	field of 4n or 12n symbols		Source Address of a frame format - Contains a fixed number of either four or twelve data quartet (n) symbols.

Name	Range	Initial value	Purpose
FC[n,n]	field of 2n symbols		Frame Control field of a PDU format - Contains two data quartet (n) symbols whose meaning are: the first data quartet n = CLFF bits and the second data quartet n = ZZZZ bits.
INFO[$n_1..n_{max}$]	field of [$n_1..n_{max}$] of symbols		Information field of a frame format - Contains a variable number of data quartet (n) symbols.
FCS[8n]	field of 8n symbols		Frame Check Sequence of a frame format - Contains eight data quartet (n) symbols.
ED[1T ∨ 2T]	field of 1T or 2T symbols		Ending Delimiter field of a PDU format - Contains one or two Terminate (T) symbols.
FS.E[S/R], FS.A[S/R], FS.C[S/R]	field of ≥ 3 S/R symbols		Frame Status field of a frame format - Contains three or more control indicator symbols Set (S) or Reset (R).
SL[4n ∨ 12n]	field of 4n or 12n symbols		Start Limit field of a subtoken format - Indicates the address of the station where the subtoken starts to be valid for use. Contains a fixed number of either four or twelve data quartet symbols.
EL[4n ∨ 12n]	field of 4n or 12n symbols		Ending Limit field of a subtoken format - Indicates the address of the station where the subtoken stops to be valid for use. Contains a fixed number of either four or twelve data quartet symbols.
CLASS[2n](*)	field of 2n symbols		Class field of a subtoken format - Indicates the duration of the frame allowed to be transmitted by the station using this subtoken. Contains a fixed number of two data quartet symbols (n).
Valid Data Length	boolean [true, false]	true	Valid Data Length- (a) Is an integral number of Data symbols pairs between SD and ED; (b) Satisfies the table of interpretation of FC field in the MAC FDDI standard.
Valid FCS_r	boolean [true, false]	true	Valid Frame Check Sequence- Satisfies the criteria of FCS checking
MSA	16-bit addresses		My Short Address
MLA	48-bit addresses		My Long Address

V. PROTOCOL VERIFICATION

A. VERIFICATION

The basic goal of a formal model for a protocol specification is to eliminate the ambiguities and difficulties associated with completeness and correctness. To avoid design errors before any implementation, logical reasoning can be applied to demonstrate that the protocol meets its specification. All possible interactions on the specified layer service are checked to ensure that the protocol satisfies the layer's specification. This is the primary goal of protocol verification.

Conceptually, protocol verification checks for general, specific, safety, and liveness properties. General properties are implicit parts of all service specification (e.g., deadlock, starvation, completeness, and termination). Specific properties require the provision of the particular service specification (e.g., the synchronous service bandwidth allotment in FDDI). Safety is the conformance verification of the protocol. It compares actual protocol actions with its service specification (e.g., if a station captures a token, it will first deliver its synchronous traffic and then its asynchronous traffic according to the timed-token rules). Liveness is a property that ensures the completion of specified services in a finite time. For analysis of protocol efficiency and responsiveness numerical bounds are considered.

Protocol correctness by the establishment of proofs is one effective approach used in protocol verification. Proofs are particularly useful for showing whether a protocol is

modeled correctly. This verification technique identifies errors and can determine the cause.

B. PROTOCOL CORRECTNESS THROUGH PROOFS

A major strength of the formal specification presented in this thesis is that it allows for proof of protocol correctness by the application of elementary logic to statements. Given a problem statement, chain-reaction arguments can be established by navigating through the state diagram and transition table. Statements are matched with enabling predicates. Each enabling predicate has the property that it must be either true or false. The statements are connected in a chain-reaction or combined to form new statements. As a result, truth tables can be constructed to prove that a statement will be either true or false on the protocol specification. Proofs can be established to verify the correctness of all protocol modules and chain-reaction arguments which consider each module precondition and postcondition can be applied to prove the complete protocol. The next subsection provides an example of proof for correctness, given a specific protocol module.

To formalize the proof the following commonly used notational conventions are assumed:

$\wedge$: and;
$\vee$: or;
$\neg$: not;
$\Rightarrow$: implies.

The chain-reaction type of argument is used, which is symbolized by the tautology $[(p \Rightarrow q) \wedge (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$.

1. Proof that an LLC Frame is Copied and an Acknowledgment is Sent

Given the following statements prove that an arriving LLC frame will be copied by the Destination Address (DA) station and an acknowledgment returned to its originator. Given the following statements:

- (a) The arriving frame has the contents of its DA field equals to one address of the set of this Station Short Addresses (SSA);
- (b) The Frame Status field (FS) has its Control Indicator "A" received as reset (R); and,
- (c) The frame is valid;

prove that in accepting the statements the MAC Receiver will copy the frame locally and return an acknowledgment.

To prove that the MAC Receiver will copy the frame and an acknowledgment will be sent it must be shown that (1) the Receiver will set the copy flag (C_Flag), (2) the Receiver will sent the frame acknowledgment signal (SIGNAL Ack_Frame) to the transmitter on the opposite ring which (3) will transmit an acknowledgment.

Proof:

In the protocol formal specification the statements (a), (b), and (c) are equivalent to $[(FC.L_r = 0) \wedge (DA_r \in \{SSA\})]$, $FS.A_r = R$, and Valid_Frame respectively. All the statements are assumed to be true, unless otherwise specified.

Let the statements be denoted as follows:

p: [(FC.L_r = 0) ∧ (DA$_r$ ∈ {SSA})],
q: FS.A_r = R,
r: Valid_Frame.

Also, denote the following statements

a: A_Flag,
s: Copy_Frame,
c: C_Flag, and
k: SIGNAL Ack_Frame
t: Tx_Idle ← clear;
PH_Request[symbol] ← Ack-buf[symbol];
symbol ← symbol + 1;
u: PH_Request(symbol) = { $PA_x[I_1..I_n]$, SD_x[J,K], FC_x[n,n], DA_x[4n or 12n], SA_x[4n or 12n], ED_x[T], FS.E_x[R/S], FS.A_x[S], FS.C_x[S] } ∧ (symbol = FS.C_x[S/R])
v: Tx_Idle ← set;

Part (1): To show that if *p* and *r*, then *c*.

The MAC Receiver State Diagram (Diag. 2) and transition table show that the statement *p* is an enabling predicate for the transition F(0-1) "DA Match". The corresponding action implied by this transition is the statement *a*. Therefore,

$$p \Rightarrow a.$$

The transition F(1-2) "Copy Frame to Local entity (LLC, SMT, MAC) shows that

$$a \Rightarrow s.$$

Given the statement *r* and the resultant logic *s* the transition F(7-8) "Frame Copied" showed in Diag. 2a, it follows that the statement *c* holds in

$$(r \wedge s) \Rightarrow c.$$

Part (2): To show that if q and c, then k.

Given q and the resultant logic of c, they will form the enabling predicate for a transition F(7-8) "Ack Frame" in Diag. 2a. of the MAC Receiver. It follows that k holds as a corresponding action of this transition, therefore

$$(q \wedge c) \Rightarrow k.$$

This part of the proof showed that the Copy_Flag will be set and the SIGNAL Ack_Frame will be sent to the transmitter on the opposite ring.

Part (3a): To show that if k then t.

The Mac Transmitter State Diagram (Diag. 9) and Transmitter Transition Table show that the statement k is an enabling predicate for the transition I0-G0 "Generate Ack." The Transmitter at this point will have the **symbol** pointer pointing to the symbol J of the SD field in the buffer array Ack-buf of symbols. The transition G(0-0) Tx SD,FC,DA,SA,ED Symbols will incrementally occur for every symbol implying the corresponding action t. Symbols from the Acknowledgment buffer are placed into the PH_indication buffer for transmission over the physical medium. Therefore,

$$k \Rightarrow t.$$

Part (3b): To show that if t then u, and if u then v.

The transition G4-I0 "Ack Issued" shows that when the symbol being transmitted out of PH_indication(symbol) buffer reaches the control indicator C of the FS field then the Transmitter has issued the acknowledgment and entered the TX IDLE state I0. For simplification, it is assumed that the pointer **symbol** is incremented and will reach the control indicator C. Note that, if no assumption is made, this step can also be

proved by induction on the array of symbols. Also, the proof could be further extended to show that the acknowledgment is sent with the control indicators E reset, A set, and C set in the frame status field FS. Therefore,

$$t \Rightarrow u$$

and finally,

$$u \Rightarrow v.$$

This part of the proof showed that the Transmitter on the opposite ring has sent an acknowledgment to the frame originator which completes the proof.

VI. CONCLUSIONS

A. REVIEW OF THE RESEARCH

The goal of the research documented in this Thesis was to develop the details at the MAC level of an improved FDDI protocol and to provide its formal specification. Essentially, the protocol improvement is achieved by increasing the ring utilization. The Thesis investigated the FDDI MAC and developed the details of the access control mechanism of a protocol that can raise the total network throughput to a maximum of 300 Megabits per second while maintaining the same data transfer of 100 Megabits per second of the original FDDI. The method developed to increase throughput was based upon simultaneous transmissions on the dual ring with concurrent ring access. A single MAC which controls access to both rings was proposed. This MAC structure adds a degree of complexity to the existing token ring FDDI network; however, with the current improvements in fiber optics technology it is feasible to design and implement such interface.

This research presented this method of access to ensure the improvement in throughput. In fact, this method called Timed-Token Controlled Concurrent Access is an enhancement to the Timed-Token Access method of FDDI which remains unchanged in the improved protocol. This method allows concurrent access to different partitions of the same physical medium; thus, increasing ring utilization. The method introduced additional PDU formats necessary to carry out the concurrent access to the ring. In

addition, changes in the FDDI stripping mechanism for protocol normal operation were necessary. Also, Algorithms to generate subtoken duration were presented and analyzed.

The improved protocol was designed to meet the requirements for a top level backbone network while maintaining key characteristics of FDDI. This backbone network can be used primarily to link lower speed Local Area Networks (LAN), including other FDDI LAN. The key characteristic of reliability offered by FDDI through its ring reconfiguration function is maintained in the improved protocol. When a failure occurs the same recovery procedures of FDDI will reconfigure the improved FDDI network. When in this condition, the network loses the improvement in throughput; however, it maintains the same functionality of FDDI, which allows communication to continue. Another key characteristic embedded in the timed-token protocol is the initialization process. This process remains unchanged in the improved protocol.

The major achievement of this thesis was the conversion of the improved FDDI protocol into a formal model of specification. The flexibility and suitability of the model "Systems of Communicating Machines" allowed the formulation of a detailed and precise protocol specification.

This specification is useful in several ways. First, it enhances the protocol understanding and interpretation. Second, it contributes to reduce protocol ambiguities which is a good feature for interoperability achievement. Third, it allows control of each protocol module or state behavior which is a desirable feature in verification and error checking. Finally, the specification can be used to proof the protocol correctness; thus, it provides means to conduct a protocol exact analysis.

B. ISSUES FOR FURTHER RESEARCH

This thesis envisioned a Local Area Network protocol that enhances the current FDDI. FDDI is already an enhancement of 802.5 token ring designed to take advantage of fiber optics using a timed mechanism that achieves simplicity and fairness. However, all the possibilities were not explored. FDDI can be improved. The resources of an FDDI network are waiting to be efficiently used. The improved FDDI protocol researched in this Thesis is a logical superset of FDDI with an added degree of complexity. Because of this complexity, only the first steps were achieved. Therefore, much work still can be done. This research opens the following areas for further studies.

- A simulation which model the improved MAC protocol.
- A comprehensive performance analysis.
- The fairness of access to transmit on the unused segment and its dependence on the Timed-Token mechanism.
- Interface implementation feasibility of the Single-MAC-Dual-PHY structure for simultaneous transmission on the dual ring with the controlled concurrent access
- Proof that the timing requirements for the subtoken can be satisfied under the timed-token access method.

The formal model for the improved protocol is a superset of FDDI. With the deletion of some transitions added for the improved protocol and small changes in the MAC Receiver and Transmitter state diagrams the formal model becomes the original FDDI or FDDI-II protocol. Conceptually, this model is an excellent tool for development of test procedures for FDDI stations interoperability achievement. Therefore, **Protocol**

Testing of FDDI, FDDI-II and the improved FDDI is also one of the research areas opened by this thesis.

LIST OF REFERENCES

1. Cidon, I., and Ofek, Y., "Metaring - A Full-duplex Ring with Fairness and Spacial Reuse," *IEEE INFOCOM'90*, Vol. 3, pp. 969-977, June 1990.

2. Jain, R., "Performance Analysis of FDDI token Ring Network: Effect of Parameters and Guidelines for Setting TTRT," *ACM SIGCOMM'90* Symposium on Communications Architecture & Protocols, pp. 264-275, September 24-27, 1990.

3. Johnson, M. J., "Proof that the Timing Requirements of the FDDI Token Ring Protocol are Satisfied," *IEEE Transactions on Communications*, Vol. COM 35, No. 6, June 1987.

4. Johnson, M. J., *Reliability Mechanisms of the FDDI High Bandwidth Token Ring Protocol*, Research Institute for Advanced Computer Science, NASA Ames Research Center, RIACS TR 85.5, July 1986.

5. Kamal, A. E., "On the Use of Multiple Tokens on Ring Networks," *IEEE INFOCOM'90*, Vol. 1, pp. 15-22, June 1990.

6. Keck, D. B., "Fundamentals of Optical Waveguide Fibers," *IEEE Communications Magazine*, Vol. 23, No. 5, May 1985.

7. Kuenzel, J., Parikh, A., and Marsh, J., "FDDI Dual Ring of Trees Topology," *Telecommunications*, October 1989.

8. Lundy, G. M., *Systems of Communicating Machines: A Model for Communication Protocols*, Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 1988.

9. Lundy, G. M., and Miller, R. E, "Specification and Analysis of a Data Transfer Protocol Using Systems of Communicating Machines," to appear in *Distributed Computing*.

10. Lundy, G. M., and Miller, R. E, "Analyzing a CSMA/CD Protocol through a Systems of Communicating Machines Specification," to appear in *IEEE Transactions on Communications*.

11. Lundy, G. M., "Improving Throughput in the FDDI Token Ring Network," *Protocols for High Speed Networks II*, North-Holland, 1991.

12. Ross, F., "An Overview of FDDI: The Fiber Distributed Data Interface," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, September 1989.

13. Ross, F., Hamstra, J., and Fink, R., "A LAN Among MANs," *Computer Communication Review*, Vol. 20, No. 3, July 1990.

14. Reinhold, B., "Recent Experience in FDDI Interoperability," *FDDI Network Design and Application*, OFC'91 Preconference Workshop, San Diego, CA, February 1991.

15. Siegel, M., "Architectural Concepts for Dual Ring Operation in FDDI," *Submitted to IFIP 3rd Conference on High Speed Networking*, Berlin, March 1991.

16. Strohl, M., "High Performance Distributed Computing in FDDI Networks," *IEEE LTS*, May, 1991.

17. Stoute, G. D., and Swanson, S. E., "How to Specify the Right FDDI Optical Fiber," *Lantimes*, February 4, 1991.

18. Working Draft Proposed American National Standard, FDDI Media Access Control (MAC-M) (Maintenance Revision), X3T9.5/88-139 X3T9/ Rev. 2.0, June 1, 1989.

19. Working Draft Proposed American National Standard, FDDI Media Access Control (MAC-2), X3T9/90-X3T9.5/88-139 Rev. 4.0, October 1990.

20. Working Draft Proposed American National Standard, FDDI Physical Layer Protocol (PHY-2), X3T9/91-X3T9.5/88-148 Rev. 4.1, March 5, 1991.

BIBLIOGRAPHY

Green, Jr. P. E., *Computer Network Architectures and Protocols*, Plenum Press, ISBN 0-306-40788-4, New York, 1982.

Hammond, J. L., and O'Reilly, P. J., *Performance Analysis of Local Computer Networks*, Addison-Wesley Publishing Company, Inc., ISBN 0-201-11530-1, March 1988.

Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI)," *IEEE Transactions on Communications*, Vol. 38, No. 8, August 1990.

Johnson, M. J., *Analysis of FDDI Synchronous Traffic Delays*, Research Institute for Advanced Computer Science, NASA Ames Research Center, RIACS TR 88.3, January, 1988.

Johnson, M. J., *Fairness of Channel Access for Non-Time-Critical Traffic Using the FDDI Token Ring Protocol*, Research Institute for Advanced Computer Science, NASA Ames Research Center, RIACS TR 86.9, March 1986.

Johnson, M. J., *Performance Analysis of FDDI*, Research Institute for Advanced Computer Science, NASA Ames Research Center, RIACS TR 88.11, April 1988.

Lundy, G. M., and Kvaslerud, O., *Applications of High Speed Networks*, M. S. Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, 1991.

Lundy, G. M., and Akyildiz, I. F., "A Formal Model of the FDDI Network Protocol," Proceedings, *EFOC/LAN'91*, London 1991.

Moore, M. V., and Oliver, V. A., "FDDI: A Federal Government LAN Solution," *Telecommunication*, Vol. 23, No. 9, September 1989.

Powers, J. P., *An Introduction to Fiber Optic Systems*, Naval Postgraduate School, September 10, 1990, To be published.

Sevick, K. C., and Johnson, M. J., "Cycle Time Properties of the FDDI Token Ring Network," *Proc. 7th Conf. on Local Computer Networks*, pp 50-56., February 1982.

Siegel, M., Sauer, K., Schodl, W., and Tangemann, M., "Design of an Enhanced FDDI-II System," *Submitted to IZS 90, "Electronic Circuits and Systems for Communication"* March 5-8, 1990, ETH Zurich.

Stallings, W., *Handbook of Computer-communications Standards*, Vol. 2, Howard W. Sams & Company, 1989.

Welzel, T., "Performance Analysis of Token Rings as High Speed Backbone Networks," *IEEE INFOCOM'90*, Vol. 1, pp. 23-29, June 1990.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA, 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943-5101	1
4.	Professor G. M. Lundy, Code CS/Ln Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
5.	Professor Amr Zaky, Code CS/ Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
6.	Diretoria de Informatica da Marinha Brazilian Naval Commission 4706 Wisconsin Ave., N.W. Washington, D.C. 20016	1
7.	Instituto de Pesquisas da Marinha Brazilian Naval Commission 4706 Wisconsin Ave., N.W. Washington, D.C. 20016	1
8.	CDR Gilberto F. Mota Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
9.	Maj. Olav Kvaslerud HQDC Norway/LST/PLAN Oslo-Mil/Huseby 0016 Oslo 1. Norway	1

10. LT Jose Luiz T. Elmiro 3
Brazilian Naval Commission
4706 Wisconsin Ave., N.W.
Washington, D.C. 20016

Thesis
E42517 Elmiro
c.1 A formal model of the
MAC layer of an improved
FDDI protocol.

4 JUN 93 80550

Thesis
E42517 Elmiro
c.1 A formal model of the
MAC layer of an improved
FDDI protocol.

3 2768 00016509 6